Compiling Dyadic First-Order Specifications into Map Algebra*

Domenico Cantone†
Andrea Formisano‡
Eugenio G. Omodeo§
Calogero G. Zarba¶

Abstract

Two techniques are designed for eliminating quantifiers from an existentially quantified conjunction of dyadic literals, in terms of the operators $\circ$, $\cap$, and $^{-1}$ of the Tarski-Chin-Givant formalism of relations. The use of such techniques is illustrated through increasingly challenging examples, and their algorithmic complexity is assessed.

Keywords: Algebraic logic, quantifier elimination, computational complexity.

1 Introduction

It was early discovered that simple algebraic specifications, which consist of listings of sort symbols, operation symbols, and equations, are in their pure form not appropriate for writing down specifications of larger software systems. Roughly speaking, in this regard they correspond to assembly code and not to structured programs of high level languages. ([8], p.3)

P. Halmos and S. Givant contend that "logic can (and perhaps should) be viewed from an algebraic perspective" (cf. [20]). This view, although quite appealing, is hard to reconcile with many practical uses of predicate logic as a wide-spectrum specification language, which appear to have little or no connection with algebra.

An indication towards unity may come from the database field. In relational DBMSs, both the data definition language and the query language are organized on two levels: SQL (or Datalog) operates at a higher and man-oriented level, while relational algebra acts as an intermediate, mainly machine-oriented, language (cf. [32, 5]).

It is conceivable that the man-machine interaction with a declarative programming system —or, more generally, with a theorem-prover or a proof-assistant— be organized similarly, with a first-order (or higher-order) predicate language interfacing man, and with the proof-search and model-building activities directly rooted on a modernized and enriched version of the Peirce-Schröder-Tarski formalism of dyadic relations (cf. [29]). Indeed, refined designs of *map algebra* —the arithmetic of dyadic relations—, and related research, constitute the most traditional and lasting effort to bridge first-order predicate reasoning with purely equational reasoning.

Of course this presupposes algorithms that perform effective translations between the two formalized levels: here is the issue we will be addressing in this paper.

*This work is partially supported by the CNR of Italy, coordinated project log(SETA) (units of Catania and of IASI/CNR, Rome), and by MURST 40%, "Tecniche formali per la specifica, l'analisi, la verifica, la sintesi e la trasformazione di sistemi software". This work is also related to the CEC-funded COST action n.274.

†Università di Catania, Dipartimento di Matematica e Informatica. cantone@dmi.unict.it
‡Università di Perugia, Dipartimento di Matematica e Informatica. formis@dipmat.unipg.it
§Università di L'Aquila, Dipartimento di Matematica Pura e Applicata. omodeo@univaq.it
¶Stanford University, Department of Computer Science. zarba@theory.stanford.edu

Various objections can be raised to the architecture just outlined. In terms of expressive power, an unquantified formalism of relations such as Tarski-Givant's map calculus (cf. [31]) corresponds to the rather limited sublanguage of first-order logic where only three individual variables are available, and only dyadic relations are taken into account. As regards deductive power, the formalism of relations is incomplete with respect to its semantics. Finally, its popularity as a specification language is somewhat low.

It should be noted, however, that a new burst of interest in the algebraic form of logic is today drifting it to unprecedented directions such as fork algebras (cf. [18]), where a pairing function with associated left and right projections is available. There is, hence, reasonable hope that the historical delay of map algebra can be recovered. The formalism of Tarski-Givant, even with no extensions (and in fact we will not extend it in this paper), can fairly compete with first-order logic when one does not simply compare the respective calculi but one instructs on the two formalisms strong theories such as theories of numbers (cf. [9, 13]), of lists and sets (cf. [22, 11]).

The main weakness of relational systems remains, in our opinion, the difficulty of a direct usage, caused by their poor readability. On the other hand, such systems "have an almost embarrassingly rich structure" —as Halmos (cf. [19]) said of Boolean algebras—, which is a reason why we believe that they can be instrumented effectively, and that sophisticated techniques for translating first-order sentences or theories into relational systems will be rewarding in not too long a run.

The main theme of the paper is how to translate dyadic first-order specifications into map algebra: some techniques aimed at that will simply be illustrated through examples (drawn from contexts as diverse as tense logic, number theory, aggregate theories, geometry), some enter into the design of translation algorithms making systematic use of them. These algorithms are *conservative*—as is unavoidable, according to a limitative result due to M. K. Kwatinetz, until conjugated projections are brought into play. This means that on occasions an algorithm may fail to effect the translation, even though the input formula is translatable.

In view of the *don't care* non-determinism ruling some of the translation actions, this situation calls for *confluence* theorems. As regards the most basic translation algorithm, confluence is proved here for the first time. More advanced algorithms exploit, for the translation, information on the single-valuedness or absoluteness of certain maps, thereby overcoming limitations of the basic approach. In such cases, the paper shows, confluence may be lost; accordingly, the order in which translating actions are performed becomes significant.

In the framework of a theory where conjugated projections are available, the translation of *any* formula can be carried out; but then, achieving a short and simple result of the translation becomes a real issue, as a few examples drawn from elementary geometry (cf. [30]) and worked out by hand suffice to show.

Above map algebra, as said at the beginning, we have in mind to place a language endowed with a higher degree of 'mental ergonomy', and hence lending itself to easier manipulations. A moral should in fact be drawn from the history of logic, which shows that the Peirce-Schröder form of logic began to vanish from the literature as soon as the Whitehead-Russell formalism, consciously modeled upon natural language, was developed.

In substitution of, or above, predicate logic —leaving out of consideration, at least momentarily, natural language— we would like to exploit a diagrammatic interface. This issue is not addressed in any depth within the paper, but some ingredients of a graphic language are beginning to appear. The paper proposes in fact a representation of map expressions based on oriented labelled graphs, which abstracts with respect to inessential features of expressions. This representation, ideal to support a description of the translation algorithms, turns out to be also useful to visualize common patterns of map reasoning (cf. [6]).

2 The Map Language and its Embedding into Predicate Logic

In this section we briefly review syntax and semantics of two languages, $\mathcal{L}^{\times}$ and $\mathcal{L}^{+}$ (cf. [31]). The former, $\mathcal{L}^{\times}$, is an equational language devoid of individual variables and quantifiers, where one can state properties of dyadic relations —MAPS, as we will call them— over an unspecified, yet fixed, *domain $\mathcal{U}$ of discourse*.

The language $\mathcal{L}^{\times}$ consists of *map equalities* $Q{=}R$, where Q and R are *map expressions*:

Definition 1 MAP EXPRESSIONS are all terms of the signature

symbol :	$\emptyset$	$\mathbb{1}$	$\boldsymbol{\iota}$	p_i	$\cap$	$\triangle$	$\circ$	$^{-1}$	$\overline{\ }$	$\setminus$	$\cup$	$\dagger$
degree :	0	0	0	0	2	2	2	1	1	2	2	2
priority :					5	3	6	7		2	2	4

where: $\emptyset$, $\mathbb{1}$ and $\boldsymbol{\iota}$ are three MAP CONSTANTS; $\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3, \ldots$ are infinitely many MAP LETTERS (whose typographical form can widely vary, e.g., $\in$, f, sval, hd, tl, $\boldsymbol{\beta}$).

The primitive constructs $\cap$, $\triangle$, $\circ$, $^{-1}$ should be intended as denoting map INTERSECTION, map SYMMETRIC DIFFERENCE, map COMPOSITION, and map CONVERSION, respectively. The signs $\overline{\ }$, $\setminus$, $\cup$, and other abbreviating constructs, will be defined in the sequel in terms of the primitives ones. All dyadic constructs will be used as left-associative infix operators, $^{-1}$ as a postfix operator, and $\overline{\ }$ as a line topping its argument. □

For an *interpretation* of $\mathcal{L}^{\times}$, one must fix a nonempty domain $\mathcal{U}$, and must put a subset $\mathsf{p}_i^{\Im}$ of the Cartesian square $\mathcal{U}^2$ in correspondence with p_i, for each $i = 1, 2, 3, \ldots$. Then each map expression P comes to designate a specific map $P^{\Im}$ (and, accordingly, any equality $Q{=}R$ between map expressions turns out to be either true or false), on the basis of the following evaluation rules:

$$
\begin{aligned}
\emptyset^{\Im} &=_{\text{Def}} \emptyset,\\
\mathbb{1}^{\Im} &=_{\text{Def}} \mathcal{U}^2,\\
\boldsymbol{\iota}^{\Im} &=_{\text{Def}} \{[a,a] \,:\, a \text{ in } \mathcal{U}\};\\
(Q{\cap}R)^{\Im} &=_{\text{Def}} \{\,[a,b] \in Q^{\Im} \,:\, [a,b] \in R^{\Im}\,\};\\
(Q{\triangle}R)^{\Im} &=_{\text{Def}} \{\,[a,b] \in \mathcal{U}^2 \,:\, [a,b] \in Q^{\Im} \text{ if and only if } [a,b] \notin R^{\Im}\,\};\\
(Q{\circ}R)^{\Im} &=_{\text{Def}} \{\,[a,b] \in \mathcal{U}^2 \,:\, \text{there are } c\text{s in } \mathcal{U} \text{ for which } [a,c] \in Q^{\Im} \text{ and } [c,b] \in R^{\Im}\,\};\\
(Q^{-1})^{\Im} &=_{\text{Def}} \{\,[b,a] \,:\, [a,b] \in Q^{\Im}\,\}.
\end{aligned}
$$

No evaluation rule is needed, of course, for derived map constructs such as the following:

$$
\begin{aligned}
\overline{P} &\equiv_{\text{Def}} \not\emptyset \equiv_{\text{Def}} P{\triangle}\mathbb{1}\,,\\
P{\cup}Q &\equiv_{\text{Def}} \overline{\overline{P}{\cap}\overline{Q}},\\
P{\setminus}Q &\equiv_{\text{Def}} P{\cap}\overline{Q},\\
P\,{\dagger}\,Q &\equiv_{\text{Def}} \overline{\overline{P}\circ\overline{Q}}.
\end{aligned}
$$

We can also extend $\mathcal{L}^{\times}$ with shortening notation for map equalities that follow certain patterns; e.g.,

$$
\begin{aligned}
P{\subseteq}Q &\equiv_{\text{Def}} P{\setminus}Q = \emptyset,\\
\mathsf{Func}(P) &\equiv_{\text{Def}} P^{-1}{\circ}P{\subseteq}\boldsymbol{\iota},\\
\mathsf{Total}(P) &\equiv_{\text{Def}} P{\circ}\mathbb{1}{=}\mathbb{1},
\end{aligned}
$$

so that, for instance:

- $\mathsf{Func}(\,P\,)$ states that $P^{\Im}$ is a partial function from $\mathcal{U}$ into $\mathcal{U}$;
- $\mathsf{Total}(\,P\,)$ states that the domain of $P^{\Im}$ is the whole $\mathcal{U}$;
- monadic predicates can be represented by map letters or expressions P subject to the condition $P{\subseteq}\boldsymbol{\iota}$.

The second language of interest, $\mathcal{L}^{+}$, is a variant version of a *first-order dyadic predicate language*: an *atomic formula* (briefly, an *atom*) of $\mathcal{L}^{+}$ has either the form xQy or the form $Q{=}R$, where x, y stand for individual variables (ranging over $\mathcal{U}$) and Q, R stand for map expressions of $\mathcal{L}^{\times}$. Propositional connectives and existential/universal quantifiers are employed as usual; atomic formulae and their negations are called *literals*. An ordering $\mathsf{v}_1, \mathsf{v}_2, \ldots$ of all individual variables is assumed. To improve readability, we will usually employ the symbol $=$ in place of $\boldsymbol{\iota}$ inside the formulae of $\mathcal{L}^{+}$; uppercase letters will often stand for individual variables ruled by universal quantifiers understood at the beginning of their formulae; moreover, the 'anonymous variable' $\_$ will represent an individual variable occurring only once inside its formula.

Example 1 The property of f being a Galois' correspondence can be specified by means of the three map equalities

$$\mathsf{f}\circ\mathsf{f}\subseteq\boldsymbol{\iota},\qquad \mathsf{f}\cap\boldsymbol{\iota}=\emptyset,\qquad \mathsf{f}^{-1}\subseteq\mathsf{f}\circ\mathbb{1},$$

as well as by the quantified sentence

$$(\,\forall x,y\,)\Big(\,\big(\,\exists z(\,x\mathsf{f}z\wedge z\mathsf{f}y\,)\rightarrow x{=}y\,\big)\wedge\big(\,x\mathsf{f}y\rightarrow x{\neq}y\,\big)\wedge\big(\,y\mathsf{f}x\rightarrow\exists v\;x\mathsf{f}v\,\big)\Big)\,.$$

□

3 Graph-based Representation of Map Expressions and Formulae

> *The arithmetical symbols are written diagrams and the geometrical figures are graphic formulas; and no mathematician could spare these graphic formulas, any more than in calculation the insertion and removal of parentheses or the use of other analytical signs.* ([21], p.5)

It is at times useful (cf. [3]) to represent a map expression P or an identity $P{=}\mathbb{1}$ or, more generally, an existentially quantified conjunction φ of literals of $\mathcal{L}^{+}$, by a directed graph whose edges are labelled by map expressions. To see the most immediate way of doing this, let us assume that φ is composed by atoms of the form xPy, where x and y are individual variables and P is a map expression. (Equality atoms $Q{=}R$ have been rewritten already in the form $x\overline{\mathbb{1}\circ(\,Q\mathbin{\triangle}R\,)\circ\mathbb{1}}y$, and negative literals in the form $x\overline{Q}y$; moreover, free variables may occur in φ intermixed with the existentially quantified variables.) A directed multi-graph G_φ representing φ is built up so that:[1]

1) G_φ has a node ν_x for each distinct variable x occurring in φ;

2) for each literal xPy in the conjunction φ, there is a labelled edge $[\nu_x, P, \nu_y]$ leading from node ν_x to node ν_y; and

3) the nodes of G_φ are subdivided into two sets: the ones that correspond to the existential variables in φ, called *bound* nodes, and all remaining nodes.

A chain of transformations can then be applied to any graph obtained in this standard fashion, by the following rules, which manifestly preserve the meaning of the graph:

[1] This paper presupposes on the part of the reader familiarity with common graph-theoretic notions and terminology (cf., e.g., [26]). In particular, by *degree* of a node ν in a graph or multi-graph G we mean the number of edges adjacent to ν in G.

1. An edge $[\nu, \mathbb{1}, \nu']$ can be removed or created between nodes ν, ν'.

2. An edge $[\nu, P, \nu']$ can be converted into $[\nu', Q, \nu]$ where either $P \equiv Q^{-\mathbf{1}}$ or $Q \equiv P^{-\mathbf{1}}$ or $P \equiv Q \equiv \boldsymbol{\iota}$.[2]

3. Two edges $[\nu, P, \nu']$ and $[\nu, Q, \nu']$ can be replaced by the single edge $[\nu, P \cap Q, \nu']$, and conversely.

4. If $[\nu, P, \nu']$ and $[\nu', Q, \nu'']$ are the only edges involving the bound node ν', they can be replaced by the single edge $[\nu, P \circ Q, \nu'']$; conversely, an edge $[\nu, P \circ Q, \nu'']$ can be replaced by two edges $[\nu, P, \nu']$ and $[\nu', Q, \nu'']$ where ν' is a new bound node.

5. An edge $[\nu, \boldsymbol{\iota}, \nu']$, where either ν' is a bound node with degree 1, or $\nu' \equiv \nu$, can be deleted; conversely, an edge $[\nu, \boldsymbol{\iota}, \nu']$ where either ν' is a new bound node or $\nu' \equiv \nu$ can be created.

6. An edge $[\nu', Q, \nu]$ (respectively, $[\nu, Q, \nu']$) can be replaced by an edge $[\nu', Q, \nu'']$ (resp., $[\nu'', Q, \nu']$) when there is an edge $[\nu, \boldsymbol{\iota}, \nu'']$ distinct from $[\nu', Q, \nu]$ (resp., from $[\nu, Q, \nu']$).

7. An isolated bound node can be deleted.

When a rule $r.$ consists of two or more possible actions, we will refer to them as $r.a$, $r.b$, etc.; e.g., $2.c$ enables one to invert the orientation of an edge labelled $\boldsymbol{\iota}$.

These basic actions can be packaged into relatively complex transformation rules, tactics, and even algorithms of some sophistication, which preserve the meaning of the representation. At the lowest level one may place, e.g.: a rule that shifts, in a single move, several edges attached to one extreme of an edge labelled $P \cap \boldsymbol{\iota}$ to the other extreme; a rule that converts $[\nu, P \cap \boldsymbol{\iota}, \nu]$ (resp., $[\nu, P \circ \boldsymbol{\iota}, \nu']$) into $[\nu, P, \nu]$ (resp., $[\nu, P, \nu']$); one that converts $[\nu, P \circ Q^{-\mathbf{1}}, \nu]$ into $[\nu, P \cap Q, \nu']$ where ν' is new and bound; etc.

At a slightly higher level, one can eliminate multiple labelled edges $[\nu, P, \nu']$ sharing the same two endpoints ν, ν', through systematic use of action $3.a$, thus reducing the multi-graph to a graph proper. At the same level, one can eliminate all loop-edges $[\nu, P, \nu]$ by introducing, for each of them, a new bound node ν' along with an edge $[\nu, P \cap \boldsymbol{\iota}, \nu']$.

A further level up, one has an algorithm for associating a planar (multi-)graph G to a given map expression P. Two designated nodes s_0 and s_1, named *source* and *sink*, will represent the two arguments of P, and every node distinct from these two will be regarded as being bound.

Algorithm *Graph fattening.* Given P, one proceeds non-deterministically to construct G, s_0, s_1, as follows: either

- G consists of a single edge, labelled P, leading from s_0 to s_1; or
- P is of the form $Q^{-\mathbf{1}}$, and G, s_1, s_0 (with source and sink interchanged) represents Q; or
- P is of the form $Q \circ R$, the disjoint graphs G', s_0, s_2' and G'', s_2'', s_1 represent Q and R respectively, and one obtains G by combination of G' with G'' by 'gluing' s_2'' onto s_2' to form a single node; or
- P is of the form $Q \cap R$, the disjoint graphs G', s_0', s_1' and G'', s_0'', s_1'' represent Q and R respectively, and one obtains G from G' and G'' by gluing s_0'' onto s_0' to form s_0 and by gluing s_1'' onto s_1' to form s_1.

(The name of this algorithm refers to the choice of resorting to the first alternative only when no other alternative is viable, so that the 'fattest' possible graph is obtained.) □

As an additional convention related to this algorithm, one can either

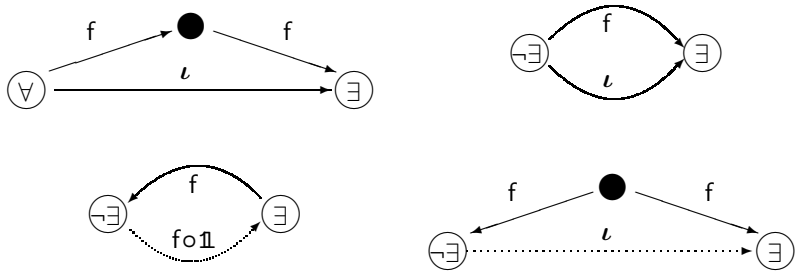
[2] Primarily, this 'conversion' is intended as an edge-replacement rule; however, it could also be intended in the sense that $[\nu', Q, \nu]$ is added to the graph without $[\nu, P, \nu']$ being removed.

Transitivity:	$X\,\delta\,Y\,\delta\,Z \;\rightarrow\; X\,\delta\,Z$	$\delta \circ \delta \subseteq \delta$
Symmetry:	$Y\,\delta\,X \;\rightarrow\; X\,\delta\,Y$	$\delta^{-1} = \delta$
Reflexivity:	$X\,\delta\,Y \;\rightarrow\; X\,\delta\,X \wedge Y\,\delta\,Y$	$\delta \cup \delta^{-1} \subseteq (\iota \cap \delta) \circ \mathbb{1}$
Antisymmetry:	$X \leq Y \leq X \;\rightarrow\; X = Y$	$\leq \cap \leq^{-1} \subseteq \iota$
Trichotomy:	$X \leq Y \vee Y \leq X$	$\leq \cup \leq^{-1} = \mathbb{1}$
Acyclicity:	$\neg\, X_0 \in X_1 \cdots \in X_n \in X_0$	$\in \circ \cdots \circ \in \cap\, \iota = \emptyset$
Density:	$X \leq Y \wedge X \neq Y \;\rightarrow\; (\exists v)(X \leq v \leq Y \wedge X \neq v \neq Y)$	$\leq \setminus \iota \subseteq (\leq \setminus \iota) \circ (\leq \setminus \iota)$
Lack of endpoints:	$(\exists v, w)(v \leq X \leq w \wedge v \neq X \neq w)$	$\iota \subseteq (\leq \setminus \iota) \circ \mathbb{1} \circ (\leq \setminus \iota)^{-1}$
Monotonicity:	$X \leq Y\,\mathsf{f}\,Z \wedge X\,\mathsf{f}\,V \;\rightarrow\; V \leq Z$	$\leq \circ\, \mathsf{f} \cap \mathsf{f} \circ \not\leq\, = \emptyset$
Bisimulation:	$\big(Y\,\beta\,X \;\rightarrow\; X\,\beta\,Y\big) \wedge \big(V\,\gamma\,X\,\beta\,Y \;\rightarrow\; (\exists w)(w\,\gamma\,Y \wedge V\,\beta\,w)\big)$	$\mathbb{1} \circ (\beta \setminus \beta^{-1}) \cup (\gamma \circ \beta \setminus \beta \circ \gamma) = \emptyset$
Graph isomorphism:	$\big((X\,\mathsf{f}\,Y \wedge X\,\mathsf{f}\,Z) \vee (Y\,\mathsf{f}\,X \wedge Z\,\mathsf{f}\,X) \;\rightarrow\; Y = Z\big) \wedge$ $\big((\exists v) X\,\mathsf{f}\,v \leftrightarrow (\exists w)(X\,\mathsf{r}\,w \vee w\,\mathsf{r}\,X)\big) \wedge \big((\exists v) v\,\mathsf{f}\,Y \leftrightarrow (\exists w)(Y\,\mathsf{s}\,w \vee w\,\mathsf{s}\,Y)\big) \wedge$ $\big(X\,\mathsf{r}\,Z \leftrightarrow (\exists v, w)(X\,\mathsf{f}\,v\,\mathsf{s}\,w \wedge Z\,\mathsf{f}\,w)\big)$	
	$\mathsf{f}^{-1} \circ \mathsf{f} \cup \mathsf{f} \circ \mathsf{f}^{-1} \subseteq \iota \;\wedge\; \mathsf{f} \circ \mathbb{1} = (\mathsf{r} \cup \mathsf{r}^{-1}) \circ \mathbb{1} \;\wedge\; \mathbb{1} \circ \mathsf{f} = \mathbb{1} \circ (\mathsf{s} \cup \mathsf{s}^{-1}) \;\wedge\; \mathsf{r} = \mathsf{f} \circ \mathsf{s} \circ \mathsf{f}^{-1}$	

Figure 1: Rosetta stone relating first-order predicate language with map language

- label both s_0 and s_1 by $\forall$, to convert a representation G, s_0, s_1 of P into a representation of the equality $P{=}\mathbb{1}$; or
- label the source by $\forall$ and the sink by $\exists$, to represent the statement $\mathsf{Total}(P)$, which is a short for $P{\circ}\mathbb{1}{=}\mathbb{1}$; or
- label both s_0 and s_1 by $\exists$, to represent the inequality $P{\neq}\emptyset$, which is a short for the equality $\mathsf{Total}(\mathbb{1}{\circ}P)$; or
- label the source by $\neg\exists$ and the sink by $\exists$, to represent the equality $P{=}\emptyset$.

An edge is usually represented in drawings by a solid arrow. By using a dotted arrow, instead, we will feel authorized to represent the associated label $\overline{P}$ simply by the writing P. Thus, for example, the following four graphs state that f is a *total* Galois' correspondence (cf. Example 1):



The above discussion does not address issues related to the map operators $\triangle$, $\cup$. Initial steps towards the treatment of these operators have been moved in [6].

4 Translating First-Order Sentences into Map Equalities

To what extent is the translation of $\mathcal{L}^+$ into $\mathcal{L}^\times$ possible (see Fig.1 for examples)?

Let us recall a definition from [31], p.62:

Definition 2 A sentence α of $\mathcal{L}^+$ is said to be EXPRESSIBLE in $\mathcal{L}^\times$ if there is a map equality $P{=}Q$ of $\mathcal{L}^\times$ for which $\alpha \equiv^+ P{=}Q$, i.e., in every interpretation $\Im$ it holds that $\alpha^\Im$ is true if and only if $P^\Im = Q^\Im$ holds. □

Among sentences expressible in this sense, one finds all sentences that involve no more than three distinct individual variables.[3]

Example 2 The axioms of MINIMAL TENSE LOGIC are (cf. [2]):

$$\begin{aligned}
&\text{i.} && [\boldsymbol{\phi}](q \to r) \to ([\boldsymbol{\phi}]q \to [\boldsymbol{\phi}]r),\\
&\text{ii.} && q \to [\boldsymbol{\phi}]\langle\boldsymbol{\pi}\rangle q,\\
&\text{iii.} && [\boldsymbol{\pi}](q \to r) \to ([\boldsymbol{\pi}]q \to [\boldsymbol{\pi}]r),\\
&\text{iv.} && q \to [\boldsymbol{\pi}]\langle\boldsymbol{\phi}\rangle q,
\end{aligned}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\pi}$ designate future and past respectively, so that $\langle\boldsymbol{\phi}\rangle$, $[\boldsymbol{\phi}]$, $\langle\boldsymbol{\pi}\rangle$, and $[\boldsymbol{\pi}]$ read "eventually", "henceforth", "sometimes in the past', and "always till now", and ii. states that *if q presently holds, it will henceforth be true that q held in the past*.

From the map-theoretic perspective, letting $\mathcal{U}$ consist of all instants, one will regard $\boldsymbol{\phi}, \boldsymbol{\pi}$ as map letters, and the propositional letters q and r as variables ranging over monadic maps (in essence, a letter designates the set of all instants when the fact it designates holds). To render the above axioms in the map language, it is convenient to regard momentarily a predicate Q as monadic iff $Q \circ \mathbb{1} = Q$.[4]

A preliminary translation into $\mathcal{L}^+$ brings i. and ii. into

$$\forall x \forall z (\neg\exists y\, x\, \boldsymbol{\phi}\, y\, Q \backslash R\, z \to \exists y\, x\, \boldsymbol{\phi}\, y\, \overline{Q}\, z \vee \neg\exists y\, x\, \boldsymbol{\phi}\, y\, \overline{R}\, z)$$

and into

$$\forall x \forall z \big(x\, Q\, z \to \neg\exists y (x\, \boldsymbol{\phi}\, y \wedge \neg\exists v\, y\, \boldsymbol{\pi}\, v\, Q\, z)\big),$$

respectively; both of these sentences are in three variables (v can in fact be renamed as x), and we easily get their following map-theoretic equivalents:

$$\begin{aligned}
&\text{i.} && \boldsymbol{\phi} \circ \overline{R} \setminus \boldsymbol{\phi} \circ \overline{Q} \subseteq \boldsymbol{\phi} \circ (Q \setminus R),\\
&\text{ii.} && Q \subseteq \overline{\boldsymbol{\phi}} \dagger \boldsymbol{\pi} \circ Q.
\end{aligned}$$

Notice that the formula i. displayed here is true for all maps $\boldsymbol{\phi}$, Q, and R. This equation, and many slight variations of it, are well known and occur in many papers that use the calculus of relations (called "map algebra" in this paper). □

One cannot do entirely without the three-variable restriction; indeed, a sentence α of $\mathcal{L}^+$ can be expressed in $\mathcal{L}^\times$ if and only if it is logically equivalent to a sentence of $\mathcal{L}^+$ in three variables. As was shown in [23], the collection of all such αs is undecidable.

In spite of the latter limitative result, it is often rather simple, in practice, to translate an $\mathcal{L}^+$-sentence into $\mathcal{L}^\times$ even if it involves more than three variables.

Conservative translation techniques that avoid moving quantifiers inwards unnecessarily were described in [3] and [14]. We briefly review in this section one of those techniques. An extension of this technique, which exploits information on the functionality of particular map symbols, will be discussed in Sec.5.

Algorithm *Graph thinning*. An existentially quantified conjunction φ of literals of the form xPy is given (cf. Sec.3).

The goal is to find a quantifier-free conjunction —or simply an atom, if there are at most two free variables in φ— equivalent to φ. Initially, a directed and labelled multi-graph G_φ representing

[3] An algorithm for translating 3-variable sentences into equations of $\mathcal{L}^\times$ was specified in detail in [31], pp.77–79—see also [13], pp.347–349 and [14], Sect. 5.

[4] This property was called 'right-absoluteness' in [14].

φ by the usual conventions is built up, then it is normalized by elimination of loop-edges, and finally it is rendered a graph by fusion of multiple edges between the same nodes.

This G_φ and its labels will be manipulated as stated below, with the aim of eliminating as many bound nodes as possible. This elimination (which represents the elimination of existential quantifiers from φ) is performed by repeatedly applying two graph-transformation rules:

BYPASS rule. Let ν be a bound node with degree 2 and let $[\nu', P, \nu]$ and $[\nu, Q, \nu'']$ be the edges adjacent to it, suitably re-oriented (by rule 2. of Sec.3) so that the former enters and the latter leaves ν. Then node ν and its edges are removed, and the new labelled edge $[\nu', P \circ Q, \nu'']$ takes their place in the graph. If an edge with endpoints ν', ν'' existed already, then, after being re-orientated to comply with the orientation $[\nu', \nu'']$, it gets fused with the new edge by the rule 3.*a* of Sec.3.

BIGAMY rule. The rule applies to a bound node ν having just one adjacent edge. Let ν, ν' be the endpoints of this edge, and assume there exist a node $\nu'' \not\equiv \nu$ and an edge with endpoints ν', ν''. Then the bigamy rule behaves as if there were an edge $[\nu, \mathbb{1}, \nu'']$ labelled $\mathbb{1}$, performing bypass of the node ν.

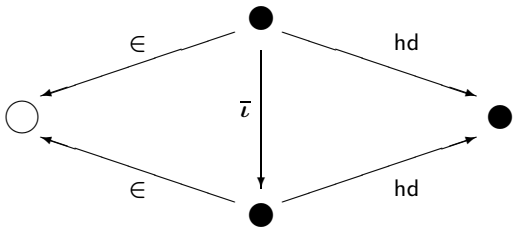
The process ends when no more applications of the previous rules are possible. If the resulting graph has no bound nodes of degree greater than 1, the sought conjunction can be directly read off the graph, else we have a failure. □

Let us see through a few examples how the above-outlined algorithm works.

Example 3 Consider the property of a set g being *single-valued* in the sense that no two elements y, z of g are ordered pairs with the same left projection hd:

$$\mathsf{sval}\, g \;\leftrightarrow_{\mathrm{Def}}\; (\forall x, y, z)(\, y \in g \,\wedge\, z \in g \,\wedge\, y\, \mathsf{hd}\, x \,\wedge\, z\, \mathsf{hd}\, x \;\rightarrow\; y = z\,).$$

The thinning algorithm, given the negation of the definiens of sval, starts by constructing the following graph,



where the white node stands for g (which is the only free variable); then it applies the bypass rule twice, ending up with the following map expression:

$$\big(\, \in^{-1} \circ (\, \overline{\iota} \cap \mathsf{hd} \circ \mathsf{hd}^{-1}\,) \cap \in^{-1}\big) \circ \mathbb{1} \cap \iota.$$

This can be simplified by means of the algebraic law

$$(\, P \cap Q^{-1}\,) \circ \mathbb{1} \cap \iota = P \circ Q \cap \iota.$$

In conclusion, the definition of sval has become

$$\mathsf{sval} \equiv_{\mathrm{Def}} \iota \backslash \in^{-1} \circ (\, \mathsf{hd} \circ \mathsf{hd}^{-1} \backslash \iota\,) \circ \in,$$

where the first occurrence of ι accounts for the fact that sval is monadic.

This example suggests a possible enhancement to the translation algorithm, ensuing from replacing the bypass rule by a more general '*cascade*' rule. The new rule would eliminate a series $\nu_0, \nu_1, \ldots, \nu_n$ of bound nodes with degree 2 whose edges, after suitable re-orientation, form a simple path $\nu'\, P_0\, \nu_0\, P_1\, \nu_1 \cdots P_n\, \nu_n\, Q\, \nu''$. The algebraic simplification law mentioned above, usable when $\nu' \equiv \nu''$, should be built into this cascade rule. □

Example 4 In [14] the following general application of graph thinning was discussed. Consider a base $\mathcal{B}$ of Horn clauses subject to the following restrictions:

- all predicate letters in $\mathcal{B}$ are dyadic;
- $\mathcal{B}$ involves no function letters, but may involve constants.

W.l.o.g., we can assume that $\mathcal{B} = \mathcal{B}_E \cup \mathcal{B}_I$, where

- the *extensional* part $\mathcal{B}_E$ of $\mathcal{B}$ is made of facts $eqe \leftarrow$, with e a constant;
- the *intensional* part $\mathcal{B}_I$ of $\mathcal{B}$ is made of clauses

$$U\, r\, V \leftarrow \bigwedge_{i=1}^{n} X_i\, p_i\, Y_i,$$

where U, V, X_i, Y_i are individual variables, U is distinct from V, $n \geq 0$, each p_i is either a map letter or $\mathsf{=}$, and r is a map letter not appearing in $\mathcal{B}_E$.

It is easy to conceive generalizations where, e.g., the letters p_i are superseded by arbitrary map expressions P_i in the body of intensional rules.

The body $\bigwedge_{i=1}^{n} X_i\, p_i\, Y_i$ of each intensional clause $U\, r\, V \leftarrow \bigwedge_{i=1}^{n} X_i\, p_i\, Y_i$ can, hence, be submitted to the algorithm described above, treating all variables but U and V as existentially bound. After rewriting every clause defining r in the form $S_j \subseteq r$ (provided the algorithm terminates with success), one can condense all such clauses into a single atom $r = \bigcup_{j=1}^{m} S_j$ (an equality, in view of [27]). □

The following example, which is a special case of the preceding one, refers to the theory of natural numbers with successor operation s and conjugated projections l(eft) r(ight) associated with Cantor's pairing function (cf. [9, 4]):

$$X, Y \mapsto \big((\, X + Y \,) \cdot (\, X + Y + 1 \,) \big)/2 + Y.$$

Example 5 Rules for evaluating l and r in the theory of successor are:

$\mathsf{l}(0,0)$	$\mathsf{r}(0,0)$
$\mathsf{l}(\mathsf{s}\,X, V) \leftarrow \mathsf{l}(X, \mathsf{s}\,V)$	$\mathsf{r}(\mathsf{s}\,X, \mathsf{s}\,\mathsf{r}\,X) \leftarrow \mathsf{l}(X, \mathsf{s}\,\_)$
$\mathsf{l}(\mathsf{s}\,X, \mathsf{s}\,\mathsf{r}\,X) \leftarrow \mathsf{l}(X, 0)$	$\mathsf{r}(\mathsf{s}\,X, 0) \leftarrow \mathsf{l}(X, 0)$

We begin by rectifying these Horn clauses into

$U\,\mathsf{l}\,V \leftarrow U\,\mathsf{z}\,V$
$U\,\mathsf{l}\,V \leftarrow X\,\mathsf{s}\,U,\ X\,\mathsf{l}\,S,\ V\,\mathsf{s}\,S$
$U\,\mathsf{l}\,V \leftarrow X\,\mathsf{s}\,U,\ X\,\mathsf{l}\,Z,\ Z\,\mathsf{z}\,\_, X\,\mathsf{r}\,R,\ R\,\mathsf{s}\,V$
$U\,\mathsf{r}\,V \leftarrow U\,\mathsf{z}\,V$
$U\,\mathsf{r}\,V \leftarrow X\,\mathsf{s}\,U,\ X\,\mathsf{l}\,S,\ \_\,\mathsf{s}\,S,\ X\,\mathsf{r}\,R,\ R\,\mathsf{s}\,V$
$U\,\mathsf{r}\,V \leftarrow X\,\mathsf{s}\,U,\ X\,\mathsf{l}\,W,\ W\,\mathsf{z}\,V$

where z represents the predicate $\{[0,0]\}$.

Through graph-thinning, one easily obtains corresponding map inclusions:

$$\mathsf{z} \subseteq \mathsf{l}, \qquad \mathsf{z} \subseteq \mathsf{r},$$
$$\mathsf{s}^{-1} \circ \mathsf{l} \circ \mathsf{s}^{-1} \subseteq \mathsf{l}, \qquad \mathsf{s}^{-1} \circ (\, \mathsf{l} \circ \mathsf{s}^{-1} \circ \mathbb{1} \cap \mathsf{r} \circ \mathsf{s}\,) \subseteq \mathsf{r},$$
$$\mathsf{s}^{-1} \circ (\, \mathsf{l} \circ \mathsf{z} \circ \mathbb{1} \cap \mathsf{r} \circ \mathsf{s}\,) \subseteq \mathsf{l}, \qquad \mathsf{s}^{-1} \circ \mathsf{l} \circ \mathsf{z} \subseteq \mathsf{r}.$$

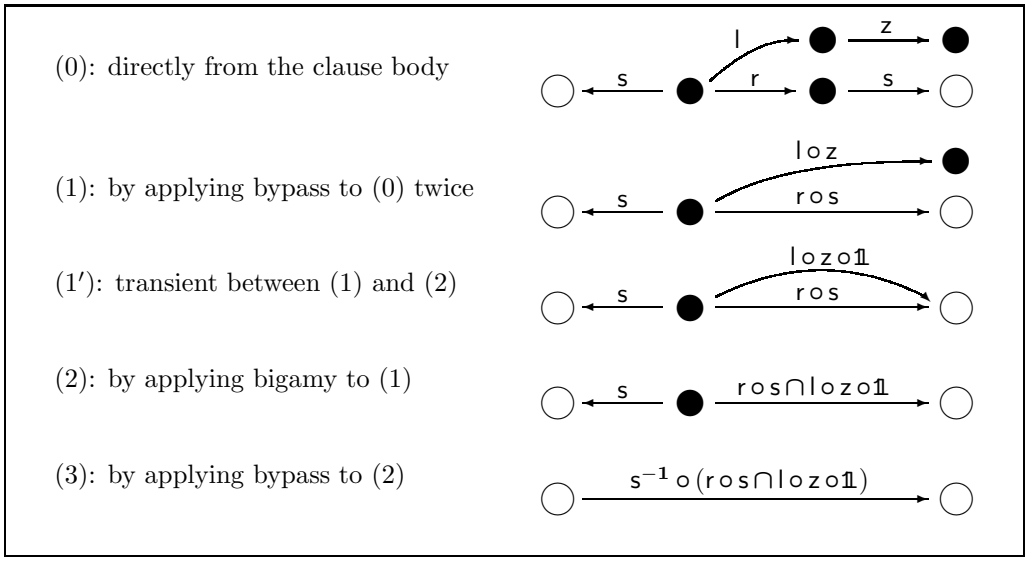


Figure 2: Steps of the translation of one of the clauses defining left selection

Fig.2 zooms in on a detail of the translation, by showing how the third map expression included in l is obtained from the body of the third clause defining l. Then we can condense the inclusions into map equalities:

$$\mathsf{l} = \mathsf{z} \triangle \mathsf{s}^{-1} \circ \mathsf{l} \circ \mathsf{s}^{-1} \triangle \mathsf{s}^{-1} \circ (\mathsf{l} \circ \mathsf{z} \circ \mathbb{1} \cap \mathsf{r} \circ \mathsf{s});$$
$$\mathsf{r} = \mathsf{z} \triangle \mathsf{s}^{-1} \circ (\mathsf{l} \circ \mathsf{s}^{-1} \circ \mathbb{1} \cap \mathsf{r} \circ \mathsf{s}) \triangle \mathsf{s}^{-1} \circ \mathsf{l} \circ \mathsf{z}.$$

(We have replaced union by symmetric difference in both equalities taking advantage of the disjointness of the operands of $\cup$.) □

The next example, as well as others that will follow, refers to a theory of nested lists generalizing the theory of flat lists in [13, 14] (hints on this more general theory are given in Fig.4).

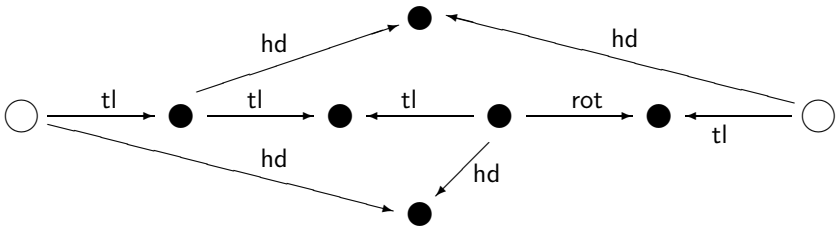
Example 6 Consider the following two clauses, which specify the operation of rotating a list:

$$\mathsf{rot}([X], [X]).$$
$$\mathsf{rot}([X, Y|Z], [Y|W]) \leftarrow \mathsf{rot}([X|Z], W).$$

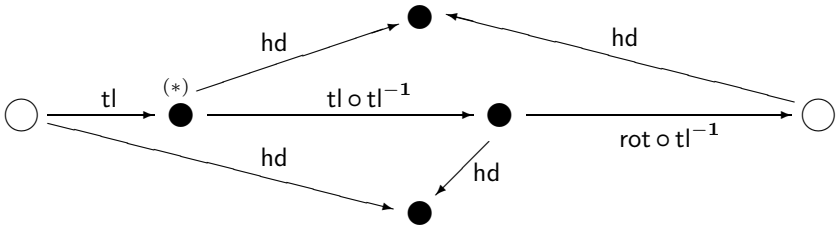
To prepare for the quantifier elimination, we rectify the clauses by resorting to list operations hd, tl (which extract the 'head' and the 'tail' of a non-null list), and nl (which checks a list for being null):

$$A\,\mathsf{rot}\,C \;\leftarrow\; A\,\mathsf{tl}\,N \wedge \mathsf{nl}\,N \wedge A = C.$$
$$A\,\mathsf{rot}\,C \;\leftarrow\; A\,\mathsf{hd}\,H \wedge A\,\mathsf{tl} \circ \mathsf{hd}\,K \wedge A\,\mathsf{tl} \circ \mathsf{tl}\,B \wedge C\,\mathsf{hd}\,K \wedge$$
$$\wedge\, C\,\mathsf{tl}\,D \wedge L\,\mathsf{hd}\,H \wedge L\,\mathsf{tl}\,B \wedge L\,\mathsf{rot}\,D.$$

It is easy to verify that starting from the body of the first clause, the above algorithm produces as output the map expression $\mathsf{tl} \circ \mathsf{nl} \circ \mathbb{1} \cap \iota$. The translation of the body of the second clause is somewhat more intricate. In this case the algorithm starts by building up the following graph (where the white nodes correspond to the variables occurring in the head of the clause):



Two applications of the bypass rule yield the graph



After two more bypass steps, the graph-thinning process is stuck, no further applications of bypass or bigamy being possible. (We will see later how to recover from this situation.) □

In Sec.4.2 we will show that the order in which the bypass or bigamy rules are applied is unimportant, in the sense that either there is no sequence of bypass or bigamy rules capable of eliminating all bound nodes, or all applicable sequences can eventually eliminate all bound nodes.

4.1 Complexity of the Thinning Algorithm

(For basic notions in algorithms and data structures, the reader is referred to [7].)

Let φ be an existentially quantified conjunction of literals of the form xPy, where x and y are individual variables and P is a map expression, and let $G_\varphi = (N_\varphi, E_\varphi)$ be the directed labelled graph representing φ.

Normalization of G_φ, obtained by fusing multiple edges between the same nodes and by eliminating loop-edges, can be performed in $\mathcal{O}(|E_\varphi| \log |E_\varphi|)$ time as follows (we assume that G_φ is represented by adjacency lists):

- fix any total ordering $<$ among the nodes in N_φ;
- sort the edges in E_φ according to the ordering $\prec$ defined by
$$[\nu, P, \nu'] \prec [\mu, Q, \mu'] \quad \text{iff} \quad \begin{array}{l} \min(\nu, \nu') < \min(\mu, \mu')\,, \text{ or} \\ (\min(\nu, \nu') = \min(\mu, \mu') \text{ and } \max(\nu, \nu') < \max(\mu, \mu')) \end{array}$$
(for the analysis below, it is also convenient to assume that, after sorting, the edges are organized as a balanced search tree);
- following the order $\prec$ among edges in E_φ, fuse multiple edges between the same nodes and eliminate loop-edges.

Let $G'_\varphi = (N'_\varphi, E'_\varphi)$ be the graph resulting after normalization: observe that $|N'_\varphi| \leq 2 \cdot |N_\varphi|$, since each elimination of a loop-edge introduces exactly one new node and only such operations can introduce new nodes.

Since each application of either the bypass or the bigamy rule has the effect to remove one node (and at least one edge) from G'_φ, it plainly follows that the algorithm obviously terminates after at most $\mathcal{O}(|N'_\varphi|) = \mathcal{O}(|N_\varphi|)$ steps.

In addition, by organizing the nodes of G_φ in a priority queue w.r.t. the degree, it is immediate to recognize that each selection of a node and adjacent edges to which to apply either the bypass or the bigamy rule can be handled in $\mathcal{O}(\log |N_\varphi|)$ time. As regards applications of the bypass rule, observe that deletion of the bypassed node together with its incident edges, and insertion of the bypassing edge (or its fusion with a pre-existing edge) can be handled in $\mathcal{O}(\log |E'_\varphi|) = \mathcal{O}(\log |N_\varphi|)$ time (we recall that the edges are maintained in a balanced search tree). Analogously, each application of the bigamy rule can be performed in $\mathcal{O}(\log |N_\varphi|)$ time. Thus, we conclude that the second phase of the algorithm, namely the one consisting in applications of the bypass and bigamy rules only, has complexity $\mathcal{O}(|N_\varphi| \log |N_\varphi|)$ time.

4.2 Confluence Property of the Graph-Thinning Algorithm

We show now that the order in which the bypass and bigamy actions are performed during an execution of the thinning algorithm is immaterial (confluence property). More specifically, we prove that given a labelled graph obtained from an existentially quantified conjunction of literals of $\mathcal{L}^+$ as explained in Section 3, either there is no sequence of bypass and bigamy steps capable of eliminating all bound nodes, or any applicable sequence eventually eliminates all bound nodes.

Edge labels and edge orientation have no influence on the success or failure of any sequence of bypass and bigamy steps. Thus, for the purpose of proving the above-stated confluence property, for the time being we can disregard both edge labels and edge orientation.

With this simplification in mind, given a bound node x in a graph G, if x has degree less than or equal to 2, then there is essentially only one way to eliminate it in one step (namely by an application of the bypass rule, if x has degree 2, or by an application of the bigamy rule, if x has degree 1). If there is indeed one, we agree that

$$\mathit{apply\_rule}(G, x)$$

denotes the graph obtained by eliminating the bound node x from G by means of the applicable rule.

The above considerations also allow us to represent any applicable sequence $\mathcal{S}$ of bypass and bigamy steps relative to a given graph G by the sequence $\langle x_1, x_2, \ldots, x_n \rangle$ of the bound nodes which are deleted from G by $\mathcal{S}$, in the order of their deletion. A sequence $\mathcal{S}$ of applicable rules is said to be *maximal* if it cannot be further extended. If the length of a maximal applicable sequence equals the number of bound nodes, then it is *successful*, otherwise it is *failing*.

Another useful observation is that an application of the bypass or the bigamy rule can only lower the degree of the nodes. Therefore, if a rule is initially applicable to a bound node x, then x will be eliminated by any maximal applicable sequence.

We are now ready to prove the following result.

Lemma 1 *Let G be a normalized graph[5] representing a given existentially quantified conjunction of literals of the form xPy. Then all of the maximal applicable sequences have the same length.*

Proof. We proceed by induction on the number of bound nodes in G.

The base case (when G contains only one bound node) is plainly true.

Concerning the inductive step, let $\mathcal{S} = \langle x_1, x_2, \ldots, x_k \rangle$ be a maximal applicable sequence of length k, and let $G' = \mathit{apply\_rule}(G, x_1)$. Notice that we can assume that $k \geq 2$, because otherwise the lemma is plainly true. By inductive hypothesis, all maximal applicable sequences relative to G' have length $k - 1$. Thus, all maximal applicable sequences relative to G and starting with the bound node x_1 have length k.

Let us assume that there exists a maximal applicable sequence relative to G of the form $\mathcal{S}' = \langle x'_1, x'_2, \ldots, x'_{k'} \rangle$, having length k' and with $x'_1 \neq x_1$.

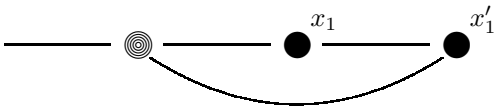
Since the degrees of x_1 and x'_1 in G both are less than or equal to 2, there must exist maximal applicable sequences $\mathcal{S}_1 = \langle x_1, x'_1, \ldots \rangle$ and $\mathcal{S}'_1 = \langle x'_1, x_1, \ldots \rangle$.

[5] I.e., loop-edges and multiple edges between the same endpoints have already been eliminated.

Let $G'' = \mathit{apply\_rule}(G, x_1')$, $G_1 = \mathit{apply\_rule}(G', x_1')$, and $G_2 = \mathit{apply\_rule}(G'', x_1)$. We show that the graphs G_1 and G_2 coincide (up to edge labels and edge orientation). We consider here only the case in which the bound nodes x_1 and x_1' are adjacent in G and leave the remaining simpler case to the reader.

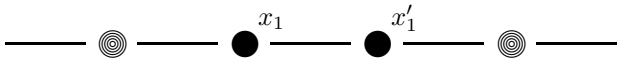
If x_1 and x_1' are adjacent, then one must consider the following two subcases (notice that in the diagrams below shaded nodes can represent bound as well as unbound nodes):

Case 1



In this case, removing x_1 by the bypass rule and then removing x_1' via the bigamy rule is equivalent to an application of the bypass rule which gets rid of x_1', followed by an application of the bigamy rule which eliminates x_1.

Case 2



It is immediate to see that the two sequences $\langle x_1, x_1' \rangle$ and $\langle x_1', x_1 \rangle$ yield the same graph. As a matter of fact, it happens that all steps are applications of the bypass rule.

The relations among the graphs G, G', G'', G_1, and G_2 are depicted in Figure 3.

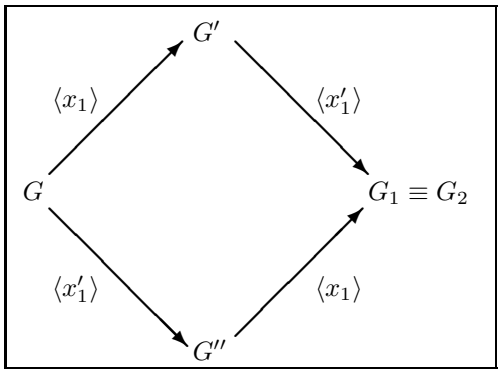


Figure 3: Relations among G, G', G'', G_1, and G_2.

Having shown that G_1 and G_2 coincide, we can now complete the proof of the lemma.

Following the path G, G', G_1, by inductive hypothesis we have that the length ℓ of each maximal applicable sequence relative to G_1 must equal $k - 2$, where we recall that k is the length of $\mathcal{S}$. On the other hand, following the path G, G'', G_2, again by inductive hypothesis we obtain that $\ell = k' - 2$, where we recall that k' is the length of $\mathcal{S}'$. Hence $k' = k$, i.e. the maximal sequences $\mathcal{S}$ and $\mathcal{S}'$ have the same length. □

An immediate consequence of the above lemma is the following corollary, which states the sought result.

Corollary 1 *Let G be a normalized graph representing a given existentially quantified conjunction of literals of the form xPy. Then all maximal applicable sequences of bypass and bigamy steps relative to G are successful, or all of them are failing.* □

5 A Refined Quantifier-Elimination Algorithm Exploiting Functionality Information

If we know that the expression P labelling a given edge $[\nu, P, \nu']$ of the graph $G_\varphi = (N_\varphi, E_\varphi)$ is single-valued, then it is possible to eliminate it by applying the following rule:

STAR rule. Let ν be a bound node with degree d greater than 2 and let $[\nu_1, P_1, \nu]$, $[\nu_2, P_2, \nu]$, ..., $[\nu_d, P_d, \nu]$ be its adjacent edges (for simplicity, we are assuming that all of them enter ν). Let us assume also that P_1 represents a single-valued map expression. Then

(a) for each $i = 2, \ldots, d$,

- we remove the edge $[\nu_i, P_i, \nu]$;
- we add the edge $[\nu_1, P_1 \circ P_i^{-1}, \nu_i]$; if an edge between ν_1 and ν_i already exists, we fuse the two edges;

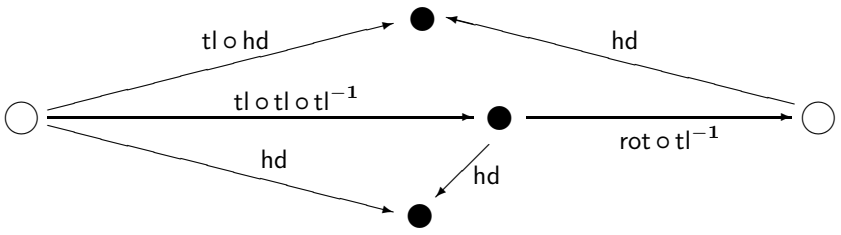
(b) we remove the edge $[\nu_1, P_1, \nu]$;

(c) we remove the bound node ν.

Correctness of the star rule follows easily from observing that it can be simulated as follows. One begins by adding $d-1$ copies $\nu'_2, \ldots, \nu'_d$ of ν and the pairs of edges $[\nu_1, P_1, \nu'_i]$ (corresponding to $[\nu_1, P_1, \nu]$) and $[\nu_i, P_i, \nu'_i]$ (corresponding to $[\nu_i, P_i, \nu]$), for each $i = 2, \ldots, d$. Because of the single-valuedness of P_1, the newly added edges just replicate the information contained in the edges $[\nu_1, P_1, \nu]$, $[\nu_2, P_2, \nu]$, ..., $[\nu_d, P_d, \nu]$, so that the latter can be removed. After reversing the edges $[\nu_i, P_i, \nu'_i]$, for $i = 2, \ldots, d$, with $d-1$ applications of the bypass rule to the pairs of adjacent edges $[\nu_1, P_1, \nu'_i]$ and $[\nu'_i, P_i^{-1}, \nu_i]$, and the subsequent deletion of the isolated nodes $\nu, \nu'_2, \ldots, \nu'_d$, one completes the simulation of the star rule.

By suitably interleaving calls to the thinning algorithm and to the star rule, sometimes it is possible to eliminate quantifiers from an existential formula φ (of the kind treated in the preceding section) for which the thinning algorithm alone fails. This is illustrated in the following example.

Example 7 Let us re-consider Example 6 in Sec.4, and assume that $\mathsf{Func}(\,\mathsf{tl}\,)$ holds. After the application of the star rule to the bound node marked $(*)$, one obtains the following graph:



By calling the thinning algorithm again on the above graph, one obtains the expression

$$(\,\mathsf{hd} \circ \mathsf{hd}^{-1} \cap \mathsf{tl} \circ \mathsf{tl} \circ \mathsf{tl}^{-1}\,) \circ \mathsf{rot} \circ \mathsf{tl}^{-1} \cap \mathsf{tl} \circ \mathsf{hd} \circ \mathsf{hd}^{-1};$$

hence, the translation of the entire specification of rot is

$$\mathsf{rot} = \big(\,\mathsf{tl} \circ \mathsf{nl} \circ \mathbb{1} \cap \iota\,\big) \triangle \big(\,(\,\mathsf{hd} \circ \mathsf{hd}^{-1} \cap \mathsf{tl} \circ \mathsf{tl} \circ \mathsf{tl}^{-1}\,) \circ \mathsf{rot} \circ \mathsf{tl}^{-1} \cap \mathsf{tl} \circ \mathsf{hd} \circ \mathsf{hd}^{-1}\,\big).$$

In [1], on top of an axiomatic specification of nested list theory (cf. [14] and Fig.4),[6] the specification of a line editor drawn from [8], p.58, was rendered in $\mathcal{L}^\times$ as shown in Fig.5. To perform the translation, the technique we are illustrating was tacitly exploited. □

$\mathsf{nl} \neq \emptyset$	$\mathsf{nl} \circ \mathbb{1} \circ \mathsf{nl} \subseteq \iota$	$\mathsf{at} \cap \mathsf{nl} = \emptyset$
$\mathsf{Func}(\,\mathsf{hd}\,)$		$\mathsf{Func}(\,\mathsf{tl}\,)$
$\mathsf{hd} \circ \mathbb{1} = \mathsf{tl} \circ \mathbb{1}$		$\mathsf{at} \triangle (\,\mathsf{nl} \circ \mathbb{1}\,) = \overline{\mathsf{tl} \circ \mathbb{1}}$
$\overline{\mathsf{at}} \subseteq \mathsf{tl}^{-1} \circ \mathsf{hd}$	$\mathsf{at} \circ \mathsf{at}^{-1} \setminus \iota \neq \emptyset$	$(\,\mathsf{hd} \circ \mathsf{hd}^{-1}\,) \cap (\,\mathsf{tl} \circ \mathsf{tl}^{-1}\,) \subseteq \iota$
		Map letters: nl, at, hd, tl

Figure 4: Weak specification of nested lists built from two or more atoms

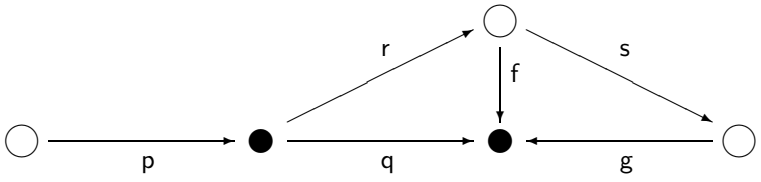
$$
\begin{array}{l}
\mathsf{rot} = \mathsf{tl} \circ \mathsf{nl} \circ \mathbb{1} \cap \iota \triangle \\
\qquad \big(\,\mathsf{hd} \circ \mathsf{hd}^{-1} \cap \mathsf{tl} \circ \mathsf{tl} \circ \mathsf{tl}^{-1}\,\big) \circ \mathsf{rot} \circ \mathsf{tl}^{-1} \cap \mathsf{tl} \circ \mathsf{hd} \circ \mathsf{hd}^{-1} \\
\mathsf{concat} = \mathsf{hd} \circ \mathsf{nl} \circ \mathbb{1} \cap \mathsf{tl} \triangle \\
\qquad \big(\,\mathsf{tl} \circ \mathsf{tl}^{-1} \cap \mathsf{hd} \circ \mathsf{tl} \circ \mathsf{hd}^{-1}\,\big) \circ \mathsf{concat} \circ \mathsf{tl}^{-1} \cap \mathsf{hd} \circ \mathsf{hd} \circ \mathsf{hd}^{-1} \\
\mathsf{char} \subseteq \mathsf{at} \cap \iota \\
\mathsf{string} = \mathsf{nl} \triangle \mathsf{hd} \circ \mathsf{char} \circ \mathbb{1} \cap \mathsf{tl} \circ \mathsf{string} \circ \mathbb{1} \cap \iota \\
\mathsf{line} = \mathsf{hd} \circ \mathsf{string} \circ \mathbb{1} \cap \mathsf{tl} \circ \mathsf{string} \circ \mathbb{1} \cap \iota \\
\mathsf{clear} = \mathsf{line} \circ \mathbb{1} \circ (\,\mathsf{hd} \circ \mathsf{nl} \cap \mathsf{tl}\,)^{-1} \\
\mathsf{createLine} = \mathsf{clear} \cap \iota \\
\mathsf{toBeginning} = \mathsf{line} \circ \mathsf{concat} \circ (\,\mathsf{hd} \circ \mathsf{nl} \circ \mathbb{1} \cap \mathsf{tl}\,)^{-1} \\
\mathsf{toEnd} = \mathsf{line} \circ \mathsf{concat} \circ (\,\mathsf{tl} \circ \mathsf{nl} \circ \mathbb{1} \cap \mathsf{hd}\,)^{-1} \\
\mathsf{delChar} = \mathsf{line} \circ \mathsf{tl} \circ \mathsf{nl} \circ \mathbb{1} \cap \iota \triangle \mathsf{line} \circ \mathsf{tl} \circ \mathsf{tl} \circ \mathsf{tl}^{-1} \cap \mathsf{hd} \circ \mathsf{hd}^{-1} \\
\mathsf{shift} = \mathsf{line} \circ \mathsf{tl} \circ \mathsf{tl} \circ \mathsf{tl}^{-1} \cap (\,\mathsf{hd} \circ \mathsf{tl}^{-1} \cap \mathsf{tl} \circ \mathsf{hd} \circ \mathsf{hd}^{-1}\,) \circ \mathsf{rot} \circ \mathsf{hd}^{-1} \\
\mathsf{moveRight} = \mathsf{line} \circ \mathsf{tl} \circ \mathsf{nl} \circ \mathbb{1} \cap \iota \triangle \mathsf{shift} \\
\mathsf{moveLeft} = \mathsf{line} \circ \mathsf{hd} \circ \mathsf{nl} \circ \mathbb{1} \cap \iota \triangle \mathsf{shift}^{-1} \\
\mathsf{addChar} = \big(\,(\,\mathsf{hd} \circ \mathsf{hd}^{-1} \cap \mathsf{tl} \circ \mathsf{tl} \circ \mathsf{tl}^{-1}\,) \circ \mathsf{tl}^{-1} \cap \mathsf{tl} \circ \mathsf{hd} \circ \mathsf{hd}^{-1}\,\big) \circ \mathsf{shift}
\end{array}
$$

Figure 5: Specification of a simple line-editor

Concerning the complexity of the above-outlined algorithm, notice that in this case it is more convenient to represent G_φ by its adjacency matrix. Thus, each call to the star rule requires $\mathcal{O}(|N_\varphi|)$ time and it has the effect of deleting one node and at least one edge. Hence, in this case, the complexity of the resulting algorithm is $\mathcal{O}(|N_\varphi|^2)$ (leaving out the time required by the normalization phase).

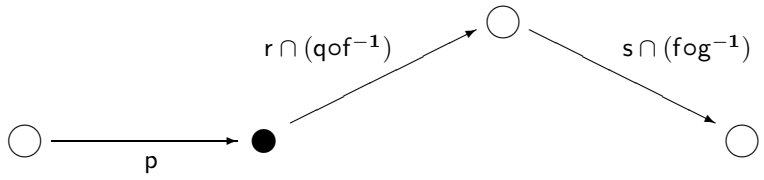
It is to be noticed, though, that the relative order in which different calls to the star rule are made can affect the outcome of the algorithm, as clarified in the following example.

Example 8 Consider the following labelled graph, where the black nodes are bound and $\mathsf{Func}(\,\mathsf{f}\,)$ and $\mathsf{Func}(\,\mathsf{g}\,)$ hold.



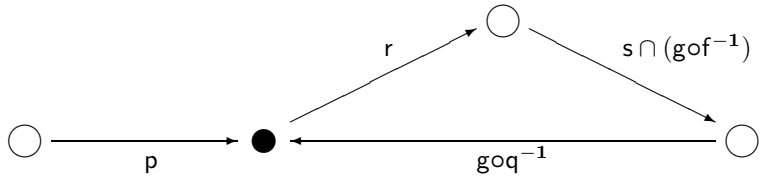
After an application of the star rule to the edge labelled by f, one obtains the following graph:

[6]The specification of lists in [14] is more sophisticated than the one in Fig.4, in that it comprises postulates of induction, acyclicity, and plenitude. In particular, plenitude ensures that infinitely many atoms intervene in the formation of lists, and we will achieve the same purpose with Example 10.



An application of the bypass rule is sufficient to eliminate the remaining bound node.

If, on the other hand, we had applied to the initial graph the star rule to the edge labelled by g, we would have obtained the graph:



Notice that the latter graph is irreducible according to our rules. □

5.1 A Further Refinement

The star rule can exploit functionality information relative to a map expression P only for edges of type $[\nu, P, \nu']$, with ν' a bound node.

When ν' is not bound, one can use instead the following more general rule:

FUNCTIONALITY rule. Let $[\nu, P, \nu']$ be a labelled edge such that P is single-valued and let $[\nu', Q, \nu'']$ be another edge, with $\nu \neq \nu''$. Then the edge $[\nu', Q, \nu'']$ is removed and the new edge $[\nu, P \circ Q, \nu'']$ is added to the graph. If the graph contains another edge between ν and ν'', then a fusion is made.

It is to be noted that in unfavorable cases the overall effect of the application of the functionality rule is just the displacement of an edge. Therefore heuristics need to be used to keep the computational complexity low.

Example 9 As an illustration of use of the functionality rule, let us consider the graph with labelled edges $[\nu_{\mathsf{v}_i}, \mathsf{p}_i, \nu_{\mathsf{v}_4}]$ $(i = 1, 2, 3)$ and $[\nu_{\mathsf{v}_1}, \mathsf{f}, \nu_{\mathsf{v}_2}]$, where ν_{v_4} is the only bound node and $\mathsf{Func}(\mathsf{f})$. Only the new transformation rule can be applied in this case, and it leads to the withdrawal of the edge $[\nu_{\mathsf{v}_2}, \nu_{\mathsf{v}_4}]$, which is replaced by the new labelled edge $[\nu_{\mathsf{v}_1}, \mathsf{f} \circ \mathsf{p}_2, \nu_{\mathsf{v}_4}]$ which, fused with the edge $[\nu_{\mathsf{v}_1}, \mathsf{p}_1, \nu_{\mathsf{v}_4}]$, yields $[\nu_{\mathsf{v}_1}, \mathsf{p}_1 \cap \mathsf{f} \circ \mathsf{p}_2, \nu_{\mathsf{v}_4}]$. One application of the bypass rule now leads to the formula $\mathsf{v}_1\ \mathsf{f}\ \mathsf{v}_2 \wedge \mathsf{v}_1\ (\ (\ \mathsf{p}_1 \cap \mathsf{f} \circ \mathsf{p}_2\ ) \circ \mathsf{p}_3^{-1}\ )\ \mathsf{v}_3$. □

6 Applications and Envisaged Enhancements

Directions for improvements of the graph-thinning algorithm can be drawn from the two examples that follow:

Example 10 A specification of lists such as the one in Fig.4, entails that $\mathsf{at} \circ \mathbb{1} = \mathsf{at}$; likewise, the definition of flat in Fig.6 enables one to prove (under an induction principle for lists that we have not explicitly stated) that $\mathsf{flat} \circ \mathbb{1} = \mathsf{flat}$. Information of this kind can be exploited to improve the quality of our quantifier-eliminating translation.

$\boldsymbol{\pi} \circ \boldsymbol{\pi}^{-1} \subseteq \boldsymbol{\iota}$		$\boldsymbol{\pi} \subseteq \mathsf{at}^{-1}$
	$\mathsf{hd} \circ \mathsf{at} \cap \mathsf{tl} \circ (\mathsf{hd} \circ \mathsf{at} \cap \mathsf{tl} \circ \mathsf{nl}) \subseteq \boldsymbol{\pi} \circ \mathbb{1}$	
$\mathsf{sameLen} = \mathsf{nl} \triangle \mathsf{tl} \circ \mathsf{sameLen} \circ \mathsf{tl}^{-1}$		$\mathsf{flat} = \mathsf{nl} \circ \mathbb{1} \triangle \mathsf{hd} \circ \mathsf{at} \cap \mathsf{tl} \circ \mathsf{flat}$
$\boldsymbol{\gamma} \subseteq \mathsf{at}^{-1}$	$\mathsf{flat} \subseteq \boldsymbol{\gamma} \circ \mathbb{1}$	$\boldsymbol{\gamma} \circ \boldsymbol{\gamma}^{-1} \cap \mathsf{sameLen} \subseteq \boldsymbol{\iota}$
		Map letters: $\boldsymbol{\pi}$, $\boldsymbol{\gamma}$, nl, hd, tl, at, sameLen, flat

Figure 6: Properties of a pairing map $\boldsymbol{\pi}$ and of a Gödel encoding map $\boldsymbol{\gamma}$

Even the simple statement that there exist distinct atoms would translate into the inequality $\mathsf{at} \circ \mathbb{1} \circ \mathsf{at}^{-1} \backslash \boldsymbol{\iota} \neq \emptyset$ instead of, more simply, into $\mathsf{at} \circ \mathsf{at}^{-1} \backslash \boldsymbol{\iota} \neq \emptyset$ if one did not take into account that at does not depend on its second argument.

The properties of a pairing map $\boldsymbol{\pi}$ and of a Gödel encoding $\boldsymbol{\gamma}$ (cf. [24], pp.41–43), can be explained roughly by saying that

- $\boldsymbol{\pi}$ associates atoms C with lists $[A, B]$ formed by two atoms so that A, B can be retrieved from C;
- $\boldsymbol{\gamma}$ associates atoms C with flat lists of atoms so that no two lists L, M of the same length fulfill $L \,\boldsymbol{\gamma}\, C$ and $M \,\boldsymbol{\gamma}\, C$ for the same C.

Formally, $\boldsymbol{\pi}$, $\boldsymbol{\gamma}$, flat and sameLen could be characterized by the clauses:

$$
\begin{array}{l}
\mathsf{sameLen}([\,],[\,]).\\
\mathsf{sameLen}([\_ \mid A],[\_ \mid B]) \leftarrow \mathsf{sameLen}(A,B).\\
\mathsf{flat}([\,],\_).\\
\mathsf{flat}([A \mid B],\_) \leftarrow \mathsf{at}(A,\_), \mathsf{flat}(B,\_).\\
\\
\mathsf{at}(A,\_) \leftarrow \boldsymbol{\pi}(\_,A).\\
X{=}Y \leftarrow \boldsymbol{\pi}(X,Z), \boldsymbol{\pi}(Y,Z).\\
\exists C\, \boldsymbol{\pi}([A,B],C) \leftarrow \mathsf{at}(A,\_), \mathsf{at}(B,\_).\\
\mathsf{at}(A,\_) \leftarrow \boldsymbol{\gamma}(\_,A).\\
L{=}M \leftarrow \boldsymbol{\gamma}(L,A), \boldsymbol{\gamma}(M,A), \mathsf{sameLen}(L,M).\\
\exists C\, \boldsymbol{\gamma}(L,C) \leftarrow \mathsf{flat}(L,\_).
\end{array}
$$

□

Example 11 The following axiom on the *betweenness* relation comes from [30], p.167:

$$\boldsymbol{\beta}(X,Y,Z) \leftarrow \boldsymbol{\beta}(X,Y,U) \wedge \boldsymbol{\beta}(Y,Z,U).$$

Our aim is, as often, to find a translation of this clause in the form of a map inclusion: $S \subseteq \boldsymbol{\beta}$ for a suitable map expression S.

Since a one-to-one correspondence exists between points and closed line segments in the Euclidean plane, we can view $\boldsymbol{\beta}$ as a dyadic relation fulfilling the clause

$$Y \,\boldsymbol{\beta}\, \underbrace{[X|Z]}_{C} \leftarrow Y \,\boldsymbol{\beta}\, \underbrace{[X|U]}_{A} \wedge Z \,\boldsymbol{\beta}\, \underbrace{[Y|U]}_{B},$$

where the interval with endpoints X, Z can be encoded by a point C, and so on. The latter clause is rectified by resorting to conjugated projections hd and tl:[7]

$$Y \,\boldsymbol{\beta}\, C \leftarrow Y \,\boldsymbol{\beta}\, A \wedge A \,\mathsf{hd} \circ \mathsf{hd}^{-1}\, C \wedge B \,\mathsf{hd}\, Y \wedge B \,\mathsf{tl} \circ \mathsf{tl}^{-1}\, A \wedge C \,\mathsf{tl}\, Z \wedge Z \,\boldsymbol{\beta}\, B.$$

[7] Our current assumptions on hd and tl are shown on top of Fig.7.

	$\mathsf{hd}^{-1} \circ \mathsf{tl} = \mathbb{1}$ $\quad$ $\mathsf{Func}(\mathsf{hd})$ $\quad$ $\mathsf{Func}(\mathsf{tl})$
	$\boldsymbol{\iota} \subseteq \mathsf{hd} \circ \mathbb{1} \circ \mathsf{tl}^{-1}$ $\quad$ $\mathsf{hd} \circ \mathsf{hd}^{-1} \cap \mathsf{tl} \circ \mathsf{tl}^{-1} \subseteq \boldsymbol{\iota}$
$\mathbf{A}_{4,6}$	$\boldsymbol{\delta}^{-1} \subseteq \boldsymbol{\delta}$ $\quad$ $\boldsymbol{\delta} \circ \boldsymbol{\delta} \subseteq \boldsymbol{\delta}$ $\quad$ $\boldsymbol{\iota} \subseteq \boldsymbol{\delta}$
	$\boldsymbol{\delta} \circ (\mathsf{hd} \circ \mathsf{tl}^{-1} \cap \mathsf{tl} \circ \mathsf{hd}^{-1}) \subseteq \boldsymbol{\delta}$
$\mathbf{A}_5$	$\boldsymbol{\delta} \circ (\mathsf{hd} \circ \mathsf{tl}^{-1} \cap \boldsymbol{\iota}) \subseteq (\mathsf{hd} \circ \mathsf{tl}^{-1} \cap \boldsymbol{\iota}) \circ \mathbb{1}$
$\mathbf{A}_1$	$\boldsymbol{\beta} \circ (\mathsf{hd} \cap \mathsf{tl}) \subseteq \boldsymbol{\iota}$
$\mathbf{A}_8$	Euclid's axiom (cf. Example 13)
$\mathbf{A}'_7$	Circle axiom (cf. Example 14)
$\mathbf{A}_{10}$	Segment construction axiom (cf. Example 15)
$\mathbf{A}_9$	Outer five-segment axiom (cf. Example 15)
$\mathbf{A}_{13}$	Elementary continuity axiom scheme (cf. Example 16)
$\mathbf{A}_{11}$	Lower dimension axiom (cf. Example 12)
$\mathbf{A}_{12}$	Upper dimension axiom (cf. Example 15)

Figure 7: Axioms on equidistance and betweeness of elementary geometry (cf. [30, 25])

Unfortunately, even the refined graph-thinning algorithm fails to find a translation S of the body of this clause. On the other hand, by transposition, we can rewrite the clause as

$$Z\,\overline{\boldsymbol{\beta}}\,B \;\leftarrow\; Y\,\boldsymbol{\beta}\,A \;\wedge\; A\,\mathsf{hd} \circ \mathsf{hd}^{-1}\,C \;\wedge\; B\,\mathsf{hd}\,Y \;\wedge\; B\,\mathsf{tl} \circ \mathsf{tl}^{-1}\,A \;\wedge\; C\,\mathsf{tl}\,Z \;\wedge\; Y\,\overline{\boldsymbol{\beta}}\,C.$$

After this change, a translation is found by the algorithm, namely

$$\Big(\big((\mathsf{hd} \circ \boldsymbol{\beta} \cap \mathsf{tl} \circ \mathsf{tl}^{-1}) \circ \mathsf{hd} \circ \mathsf{hd}^{-1} \cap \mathsf{hd} \circ \overline{\boldsymbol{\beta}}\big) \circ \mathsf{tl}\Big)^{-1} \subseteq \overline{\boldsymbol{\beta}}.$$

□

The following five examples, which also refer to the betweenness relation in elementary geometry (cf. [25], pp.95–96) suggest ways in which the translation technique could be enhanced when conjugated projections hd, tl as in the preceding example are available.

Example 12 The *lower dimension axiom* for plane geometry states that

$$(\exists\, X, Y, Z)(\neg X\;\boldsymbol{\beta}\,[Y|Z] \wedge \neg Y\;\boldsymbol{\beta}\,[Z|X] \wedge \neg Z\;\boldsymbol{\beta}\,[X|Y])\,.$$

To prepare for the translation, we restate the axioms exclusively in terms of line segments, taking $A \equiv [X|Y]$, $B \equiv [Y|Z]$, and $C \equiv [Z|X]$:

$$\begin{aligned}(\exists\, A, B, C)(A\,\mathsf{hd} \circ \overline{\boldsymbol{\beta}}\,B \;\wedge\; B\,\mathsf{hd} \circ \overline{\boldsymbol{\beta}}\,C \;\wedge\; C\,\mathsf{hd} \circ \overline{\boldsymbol{\beta}}\,A \;\wedge\; A\,\mathsf{tl} \circ \mathsf{hd}^{-1}\,B \\ \wedge\, B\,\mathsf{tl} \circ \mathsf{hd}^{-1}\,C \;\wedge\; C\,\mathsf{tl} \circ \mathsf{hd}^{-1}\,A)\,.\end{aligned}$$

Then we obtain the translation through graph-thinning:

$$(\mathsf{hd} \circ \overline{\boldsymbol{\beta}} \cap \mathsf{tl} \circ \mathsf{hd}^{-1}) \circ (\mathsf{hd} \circ \overline{\boldsymbol{\beta}} \cap \mathsf{tl} \circ \mathsf{hd}^{-1}) \cap \mathsf{hd} \circ \mathsf{tl}^{-1} \cap (\mathsf{hd} \circ \overline{\boldsymbol{\beta}})^{-1} \neq \emptyset.$$

□

Example 13 *Euclid's axiom* states that

$$\begin{aligned}\big(\forall\, U, V, W, X, Y\big)\big(W\;\boldsymbol{\beta}\,[U|Y] \wedge W\;\boldsymbol{\beta}\,[V|X] \wedge W \neq U \;\rightarrow \\ (\exists\, Z, T)(V\;\boldsymbol{\beta}\,[U|Z] \wedge X\boldsymbol{\beta}[U|T] \wedge Y\;\boldsymbol{\beta}\,[Z|T])\big)\,.\end{aligned}$$

As before, a convenient preparatory step is to package some of the point variables into segment variables; taking $A \equiv [U|Y]$, $B \equiv [V|X]$, $L \equiv [U|Z]$, $C \equiv [U|T]$, and $D \equiv [Z|T]$, we can restate the axiom as follows:

$$\big(\forall\, A, B\big)\Big(\, A\,(\,\boldsymbol{\beta}^{-1}\backslash \mathsf{hd}\,)\circ\boldsymbol{\beta}\, B \;\rightarrow\; (\,\exists\, U, L, C, D\,)\big(\, A\, \mathsf{hd}\, U \wedge C\, \mathsf{hd}\, U \wedge A\, \mathsf{tl}\circ\boldsymbol{\beta}\, D \wedge \\ D\, \mathsf{tl}\circ\mathsf{tl}^{-1}\, C \wedge C\,(\,\mathsf{tl}\circ\boldsymbol{\beta}\,)^{-1}\, B \wedge D\, \mathsf{hd}\circ\mathsf{tl}^{-1}\, L \wedge L\,(\,\mathsf{hd}\circ\boldsymbol{\beta}\,)^{-1}\, B \wedge L\, \mathsf{hd}\, U\,\big)\Big)\,.$$

Unfortunately, the consequent of this implication cannot be reduced by the thinning algorithm, and we must restructure it by calling the quadruple $[[U|T], [Z|T]]$ into play. It only requires a little ingenuity, with this trick in mind, to translate Euclid's axiom into

$$(\,\boldsymbol{\beta}^{-1}\backslash\mathsf{hd}\,)\circ\boldsymbol{\beta}\subseteq\big(\,\mathsf{hd}\circ(\,\mathsf{hd}\circ\mathsf{hd}\,)^{-1}\cap\mathsf{tl}\circ\boldsymbol{\beta}\circ\mathsf{tl}^{-1}\,\big)\circ\big(\,\boldsymbol{\iota}\cap\mathsf{hd}\circ\mathsf{tl}\circ(\,\mathsf{tl}\circ\mathsf{tl}\,)^{-1}\,\big)\circ \\ \big(\,\mathsf{hd}\circ(\,\mathsf{tl}\circ\boldsymbol{\beta}\,)^{-1}\cap(\,\mathsf{tl}\circ\mathsf{hd}\circ\mathsf{tl}^{-1}\cap\mathsf{hd}\circ\mathsf{hd}\circ\mathsf{hd}^{-1}\,)\circ(\,\mathsf{hd}\circ\boldsymbol{\beta}\,)^{-1}\,\big)\,.$$

□

Example 14 The *inner Pasch axiom* states that

$$\big(\forall\, U, V, W, X, Y\,\big)\big(\, V\,\boldsymbol{\beta}\,[U|W] \wedge X\,\boldsymbol{\beta}\,[Y|W] \;\rightarrow\; (\,\exists\, Z\,)(\, Z\,\boldsymbol{\beta}\,[V|Y] \wedge Z\,\boldsymbol{\beta}\,[X|U]\,)\,\big)\,.$$

With the help of a few variables standing for suitable line segments and a triple of points, namely $D \equiv [X|U]$, $A \equiv [U|W]$, $B \equiv [Y|W]$, and $T \equiv [[V|Y]|W]$, we can restate this axiom as follows:

$$\big(\forall\, T, D\,\big)\big(\\ (\,\exists\, B\,)(\, T\,\mathsf{tl}\circ\mathsf{tl}^{-1}\, B \wedge T\,\mathsf{hd}\circ\mathsf{tl}\circ\mathsf{hd}^{-1}\, B \wedge D\,\mathsf{hd}\circ\boldsymbol{\beta}\, B\,)\wedge \\ (\,\exists\, A\,)(\, T\,\mathsf{tl}\circ\mathsf{tl}^{-1}\, A \wedge D\,\mathsf{tl}\circ\mathsf{hd}^{-1}\, A \wedge T\,\mathsf{hd}\circ\mathsf{hd}\circ\boldsymbol{\beta}\, A\,) \\ \rightarrow\; (\,\exists\, Z\,)(\, T\,\mathsf{hd}\circ\boldsymbol{\beta}^{-1}\, Z \wedge Z\,\boldsymbol{\beta}\, D\,)\big)\,.$$

It is then straightforward to reach the following quantifier-free formulation:

$$(\,\mathsf{tl}\circ\mathsf{tl}^{-1}\cap\mathsf{hd}\circ\mathsf{tl}\circ\mathsf{hd}^{-1}\,)\circ(\,\mathsf{hd}\circ\boldsymbol{\beta}\,)^{-1}\cap(\,\mathsf{tl}\circ\mathsf{tl}^{-1}\cap\mathsf{hd}\circ\mathsf{hd}\circ\boldsymbol{\beta}\,)\circ\mathsf{hd}\circ\mathsf{tl}^{-1}\subseteq\mathsf{hd}\circ\boldsymbol{\beta}^{-1}\circ\boldsymbol{\beta}\,.$$

It is reported in [28] that in axiomatic sistems for elementary geometry which comprise continuity (see Example 16 below) the Pasch axiom can be interchanged with the following *circle axiom*:

$$(\forall\, X, Y, Z, W\,)\big(\, X\,\boldsymbol{\beta}\,[Y \mid Z] \;\rightarrow\; (\,\exists\, V\,)(\, X\,\boldsymbol{\beta}\,[V \mid W] \;\wedge\; [Y \mid V]\,\boldsymbol{\delta}\,[Y \mid Z]\,)\,\big).$$

The latter can easily be translated into the map equality

$$\mathsf{hd}\circ\boldsymbol{\beta}\setminus\mathsf{hd}\circ(\,\mathsf{hd}^{-1}\cap\boldsymbol{\beta}\circ\mathsf{tl}^{-1}\,)\circ\mathsf{tl}\circ\mathsf{hd}\circ\mathsf{tl}^{-1}\circ(\,\mathsf{hd}\circ\mathsf{hd}^{-1}\cap\boldsymbol{\delta}\,)=\emptyset.$$

□

Example 15 The *segment construction axiom* states that

$$(\forall\, X, Y, U, V\,)(\,\exists\, Z\,)(\, Y\,\boldsymbol{\beta}\,[X|Z] \;\wedge\; [U|V]\,\boldsymbol{\delta}\,[Y|Z]\,)\,,$$

where $\boldsymbol{\delta}$ designates *equidistance*, so that $[U|V]\,\boldsymbol{\delta}\,[Y|Z]$ means that the distance between U an V equals the distance between Y and Z. Once more, the difficulty of the translation is in finding a good way of packaging variables. After putting $A \equiv [X|[Y|[U|V]]]$, $B \equiv [X|Z]$, and $C \equiv [Y|Z]$, it is easy to get to the translation

$$\Big(\,\big(\,(\,\mathsf{tl}\circ\mathsf{hd}\circ\boldsymbol{\beta}\cup\overline{\mathsf{hd}\circ\mathsf{hd}^{-1}}\,)\cap(\,\mathsf{tl}\circ\mathsf{tl}\circ\boldsymbol{\delta}\cup\overline{\mathsf{tl}\circ\mathsf{hd}\circ\mathsf{hd}^{-1}}\,)\,\big)\dagger\overline{\mathsf{tl}}\,\Big)\circ\mathbb{1}=\mathbb{1}.$$

Here, roughly speaking, $\dagger$ stands for an element of the form $[\_|Z]$, so that clearly $\mathsf{tl}\circ\mathsf{hd}\circ\boldsymbol{\beta}\cup\overline{\mathsf{hd}\circ\mathsf{hd}^{-1}}$ translates $Y\,\boldsymbol{\beta}\,[X|Z]$ and $\mathsf{tl}\circ\mathsf{tl}\circ\boldsymbol{\delta}\cup\overline{\mathsf{tl}\circ\mathsf{hd}\circ\mathsf{hd}^{-1}}$ translates $[U|V]\,\boldsymbol{\delta}\,[Y|Z]$.

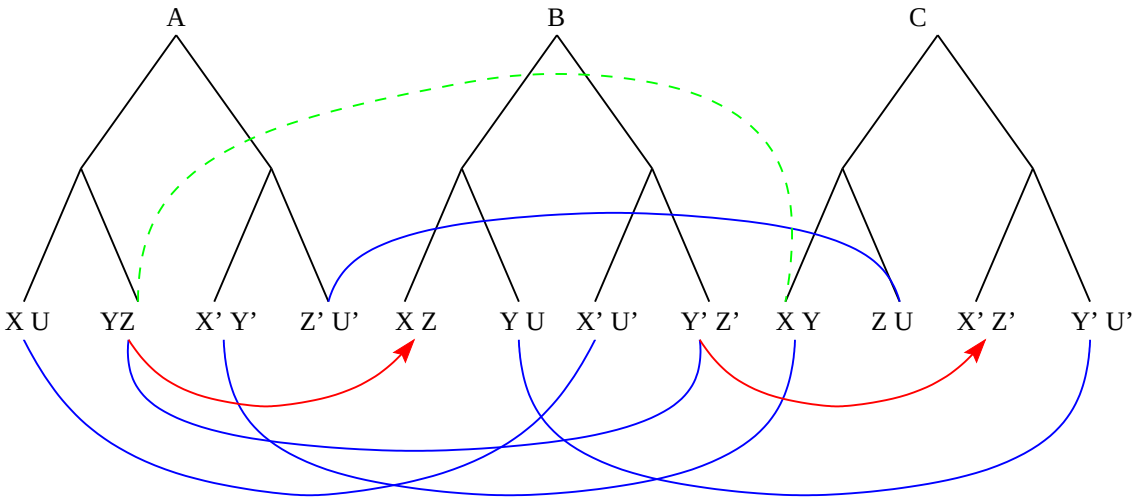


Figure 8: Packaging of variables in the five-segment axiom

The difficulties with the *outer five-segment axiom*, which states that

$$(\forall X,Y,Z,U,X',Y',Z',U')([X|Y]\,\boldsymbol{\delta}\,[X'|Y']\ \wedge\ [Y|Z]\,\boldsymbol{\delta}\,[Y'|Z']\ \wedge\ [X|U]\,\boldsymbol{\delta}\,[X'|U']\\ \wedge\ [Y|U]\,\boldsymbol{\delta}\,[Y'|U']\ \wedge\ Y\,\boldsymbol{\beta}\,[X|Z]\ \wedge\ Y'\,\boldsymbol{\beta}\,[X'|Z']\ \rightarrow\ X{=}Y\ \vee\ [Z|U]\,\boldsymbol{\delta}\,[Z'|U'])\,,$$

are similar, but magnified by the size of the sentence.

A convenient way of packaging variables, in this case, is to put

$$\begin{aligned} A &\equiv [[[X|U]|[Y|Z]]|[[X'|Y']|[Z'|U']]]\,,\\ B &\equiv [[[X|Z]|[Y|U]]|[[X'|U']|[Y'|Z']]]\,,\\ C &\equiv [[[X|Y]|[Z|U]]|[[X'|Z']|[Y'|U']]] \end{aligned}$$

(cf. Fig.8). Then one straightforwardly restates the axiom in the form

$$(\forall A,B,C)\big(A\,(P_8{\cap}P_3)\,B\,(Q_8{\cap}Q_2)\,C\ \rightarrow\ A\,R_3\,C\big)\,,$$

which translates into $\overline{(P_8{\cap}P_3)\circ(Q_8{\cap}Q_2)}\cup R_3 = \mathbb{1}$. Here P_8 is to reflect the fact that the eight ultimate constituents of A and B are the same, and shows how they are rearranged; the role of Q_8, which relates B with C, is similar; P_3 encodes the two $\boldsymbol{\delta}$-literals and the $\boldsymbol{\beta}$-literal that relate A with B; Q_2 encodes the $\boldsymbol{\delta}$-literal and the $\boldsymbol{\beta}$-literal that relate B with C; finally, R_3 encodes the literals that relate A with C. Entering into some detail, we have:

$$\begin{aligned} P_3 &\equiv \mathsf{hd}\circ\mathsf{tl}\circ\boldsymbol{\delta}\circ(\mathsf{tl}\circ\mathsf{tl})^{-1}\cap\mathsf{hd}\circ\mathsf{hd}\circ\boldsymbol{\delta}\circ(\mathsf{tl}\circ\mathsf{hd})^{-1}\cap\mathsf{hd}\circ\mathsf{tl}\circ\mathsf{hd}\circ\boldsymbol{\beta}\circ(\mathsf{hd}\circ\mathsf{hd})^{-1}\,,\\ Q_2 &\equiv \mathsf{tl}\circ\mathsf{tl}\circ\mathsf{hd}\circ\boldsymbol{\beta}\circ(\mathsf{tl}\circ\mathsf{hd})^{-1}\cap\mathsf{hd}\circ\mathsf{tl}\circ\boldsymbol{\delta}\circ(\mathsf{tl}\circ\mathsf{tl})^{-1}\,,\\ R_3 &\equiv \mathsf{tl}\circ\mathsf{hd}\circ\overline{\boldsymbol{\delta}}\circ(\mathsf{hd}\circ\mathsf{hd})^{-1}\cap\mathsf{tl}\circ\mathsf{tl}\circ\boldsymbol{\delta}\circ(\mathsf{hd}\circ\mathsf{tl})^{-1}\cap\mathsf{hd}\circ\mathsf{tl}\circ\mathsf{hd}\circ(\mathsf{hd}\circ\mathsf{hd}\circ\mathsf{hd})^{-1}\,, \end{aligned}$$

etc.

The *upper dimension axiom*, states (for two-dimensional space) that

$$(\forall U,V,W,X,Y)([U|X]\,\boldsymbol{\delta}\,[U|Y]\ \wedge\ [V|X]\,\boldsymbol{\delta}\,[V|Y]\ \wedge\ [W|X]\,\boldsymbol{\delta}\,[W|Y]\ \rightarrow\\ X{=}Y\ \vee\ V\,\boldsymbol{\beta}\,[U|W]\ \vee\ W\,\boldsymbol{\beta}\,[V|U]\ \vee\ U\,\boldsymbol{\beta}\,[W|V]).$$

This axiom is mildly simpler than the above five-segment axiom, and can be tamed by the same technique just used, with

$$\begin{aligned} A &\equiv [[[W|Y]|[U|X]]|[W|V]],\\ B &\equiv [[[U|Y]|[V|X]]|[U|W]],\\ C &\equiv [[[V|Y]|[W|X]]|[V|U]]. \end{aligned}$$

□

Example 16 The *elementary continuity axiom scheme* is as follows:

$$(\forall\,\vec{V})\Big(\,(\exists\,Z)(\forall\,X,Y)\big(\,\varphi(X,\vec{V})\wedge\psi(Y,\vec{V})\;\rightarrow\;X\,\boldsymbol{\beta}\,[Z|Y]\,\big)\;\rightarrow\\ (\exists\,Z)(\forall\,X,Y)\big(\,\varphi(X,\vec{V})\wedge\psi(Y,\vec{V})\;\rightarrow\;Z\,\boldsymbol{\beta}\,[X|Y]\,\big)\,\Big)\,.$$

Carrying out the translation in this case is straightforward, provided X, Y are packaged into a single variable, and the list $\vec{V}$ of variables is likewise treated as a single variable, so that φ and ψ can be replaced by map variables P, Q. The resulting scheme simply is

$$\emptyset\,\dagger\,\overline{\boldsymbol{\beta}}\circ(\,\mathsf{hd}\circ P\cap\mathsf{tl}\circ Q\,)\subseteq\emptyset\,\dagger\,\mathsf{hd}^{-1}\circ(\,\overline{\mathsf{hd}\circ\boldsymbol{\beta}}\cap\mathsf{tl}^{-1}\circ\mathsf{tl}\,)^{-1}\circ(\,\mathsf{hd}\circ P\cap\mathsf{tl}\circ Q\,),$$

where —so-to-say— the first $\circ$ on the right of $\subseteq$ represents the segment $[Z|Y]$. □

An ambitious goal is to enhance the quantifier elimination technique of this paper so that it can handle formulae with arbitrary nesting of quantifiers (both existential and universal), so as to achieve a good balance between efficiency and range of applicability. The following example is meant to give an indication of the kind of generalization we are aiming at.

(E)	$\not\ni\dagger\in\cap\ni\dagger\notin\subseteq\boldsymbol{\iota}$
(N)	$\overline{\mathbb{1}\circ\in}\circ\mathbb{1}=\mathbb{1}$
(WL)	$\Big(\,(\,\in\circ\in\setminus\overline{\boldsymbol{\iota}}\circ(\,\in\circ\in\cap\notin\circ\in\,)\,)\cap\notin\circ\in\,\Big)\circ\ni=\mathbb{1}$
(R)	$\mathbb{1}\circ\big(\,\overline{\mathbb{1}\circ\in}\cup\in\setminus\ni\circ\in\,\big)=\mathbb{1}$

Figure 9: Axioms of a weak membership theory

Example 17 A weak theory of sets can be based on the axioms of *extensionality*, *null set*, single-element *adjunction* and *removal*, and *regularity*, which are:

$$\begin{aligned}
&(\mathbf{E}) && \forall\,v\,(\,v\in X\;\leftrightarrow\;v\in Y\,)\rightarrow X=Y,\\
&(\mathbf{N}) && \exists\,z\,\forall\,v\;v\notin z,\\
&(\mathbf{W}) && \exists\,w\,\forall\,v\,(\,v\in w\;\leftrightarrow\;v\in X\;\vee\;v=Y\,),\\
&(\mathbf{L}) && \exists\,\ell\,\forall\,v\,(\,v\in\ell\;\leftrightarrow\;v\in X\wedge v\neq Y\,),\\
&(\mathbf{R}) && \exists\,r\,\forall\,v\,\big(\,(\,v\in X\rightarrow r\in X\,)\wedge\neg(\,v\in r\wedge v\in X\,)\,\big).
\end{aligned}$$

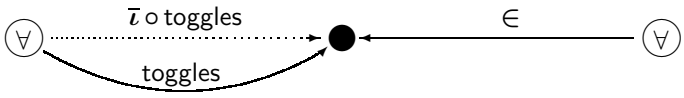
Neither of **(W)** and **(L)** is, when taken alone, expressible in $\mathcal{L}^{\times}$ (cf. [23]); however, one notices that taken together with **(N)** these axioms enable one to build the pair

$$\big\{\,Y\cup\{X\},\,Y\setminus\{X\}\,\big\}$$

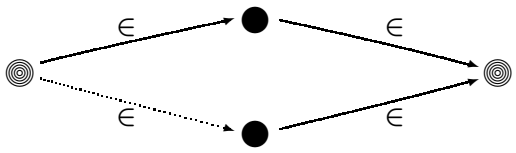
out of given sets X and Y. The latter fact, thanks to **(E)**, can be stated as

$$(\exists\,d,\,v,\,w\,)\big(\quad Y\in d\;\wedge\;X\in v\in d\;\wedge\;X\notin w\in d\;\wedge\\ (\forall\,u,\,v,\,w\,)(\,u\in v\in d\;\wedge\;u\notin w\in d\;\rightarrow\;u=X\,)\quad\big)\,,$$

and in turn (again with the contribution of **(E)**) it yields **(N)**, **(W)**, and **(L)**. This statement is rendered by the graph



inside which one has the following 'grafting' of toggles:



Since map expressions on dotted edges must always be complemented, in particular $\mathsf{toggles} \equiv_{\text{Def}} \in \circ \in \cap \not\in \circ \in$.

Globally, the set axioms under discussion can hence be translated as shown in Fig.9, where $\ni \equiv_{\text{Def}} \in^{-1}$. □

7 Conclusions

After exploratory implementations of the graph-thinning algorithm, carried out in SETL and in Java (cf. [14]), we chose to exploit a general graph-rewriting system for an implementation which one could quickly extend incrementally. A prototype translator encompassing bypass, bigamy, and the functionality rule has been achieved with AGG (acronym for 'Attributed Graph Grammars'), a tool for algebraic graph transformation developed at the TU, Berlin (cf. [10]): the screen-shot in Fig.10 was taken during a graph-thinning session with AGG and shows an application of the functionality rule during the translation of Examples 6 and 7. Additional work is needed in order to make the translation process, which still relies on user guidance, fully automatic; moreover, we count on enhancements of AGG (which are along the way) for a straightforward implementation of composite rules such as the cascade rule (see Example 3) and the star rule, and for the specification of sophisticated rule-activation patterns.

Notice that two fully automatic processes for translating *all* first-order sentences into equations of map algebra containing hd and tl, as is done (by manual treatment) in Examples 6—7 and 10–16, and in Figures 4 and 5, are explicitly constructed by [31], on pp.107–114. It would be worthwhile to implement these algorithms in some computer software system.

The main purpose of translating dyadic first-order sentences into the map language is to subsequently carry out proof-search and model-building activities either by means of standard equality-reasoning tools, or by a specialized system for map reasoning, applied to the resulting equations. As an example of the former approach, a map rendering of the axioms of ZF —the classical Zermelo-Fraenkel set theory— was carefully designed in [11], and a series of experiments with this equational formulation of ZF was then started (cf. [15, 16]). On the other hand, developing specialized tools for map reasoning does not presuppose necessarily that one works with a traditional representation of map expressions and equalities: the graph-based representation used in this paper can directly support the implementation of inference steps through graph-rewriting (cf. [6, 12, 17]): by way of example, see Fig.11.

Acknowledgements

The authors would like to thank Monica Iezzi and Annalisa Chiacchiaretta, who helped translating into map algebra the axioms of tense logic and of elementary geometry.

Olga Runge, Gabriele Taentzer, and Thorsten Schultzke provided kind support and invaluable suggestions about the AGG system.

References

[1] Aureli, F., Formisano, A., Omodeo, E. G., and Temperini, M. (1998). Map calculus: Initial application scenarios and experiments based on Otter. Technical Report 466, IASI-CNR.

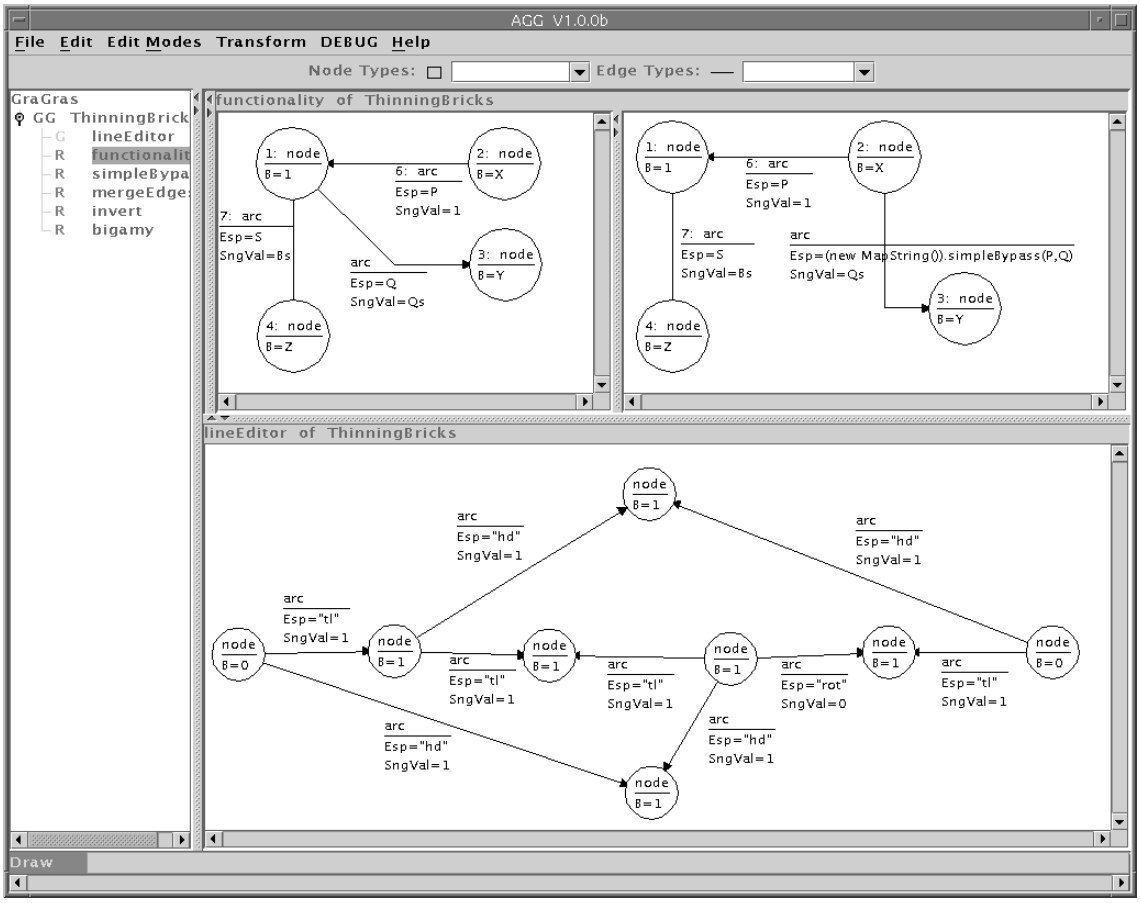


Figure 10: The AGG user interface displaying a graph-thinning session

[2] Burgess, J. (1984). Basic tense logic. In Gabbay, D. M. and Guenthner, F., editors, *Handbook of Philosophical Logic*, volume II, pages 89–133. D. Reidel, Dordrecht-Holland.

[3] Cantone, D., Cavarra, A., and Omodeo, E. G. (1997). On existentially quantified conjunctions of atomic formulae of $\mathcal{L}^+$. In Bonacina, M. P. and Furbach, U., editors, *Proc. of the FTP97 Int. Workshop on First-Order Theorem Proving*. RISC-Linz Report Series No.97-50:45–52.

[4] Cegielski, P. and Richard, D. (2001). Decidability of the theory of the natural integers with the Cantor pairing function and the successor. *Theoretical Computer Science*, 257:51–77.

[5] Ceri, S., Gottlob, G., and Tanca, L. (1990). *Logic Programming and Databases*. Surveys in Computer Science. Springer-Verlag.

[6] Chiacchiaretta, A., Formisano, A., and Omodeo, E. G. (2000). Map reasoning through existential multigraphs. Technical Report 5, Dipartimento di Matematica Pura ed Applicata, Università di L'Aquila.

[7] Cormen, T., Leiserson, C., and Rivest, R. (1990). *Introduction to Algorithms*. MIT Press.

[8] Ehrig, H. and Mahr, B. (1985). *Fundamentals of algebraic specification 1 – Equations and initial semantics*. Monographs on Theoretical Computer Science. Springer-Verlag, Berlin.

[9] Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press, New York and London.

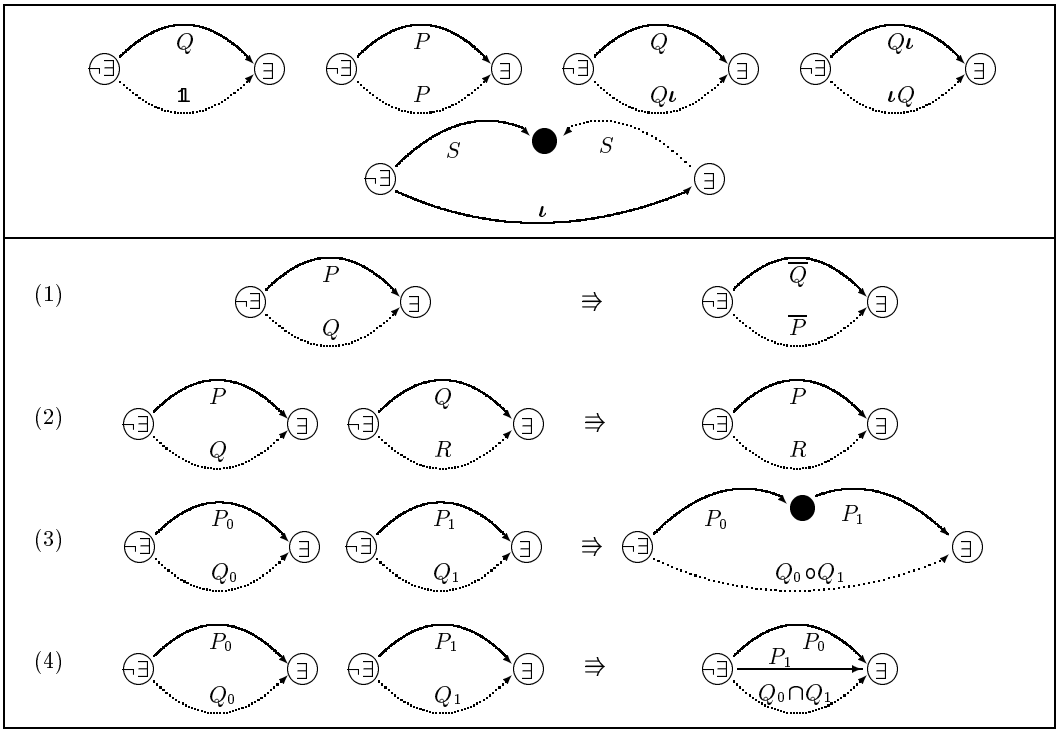


Figure 11: Five axioms and four inference rules for map inclusion

[10] Ermel, C., Rudolf, M. and Taentzer G. (1999). The AGG Approach: Language and Environment. In Ehrig, H., Engels, G., Kreowski, H.-J., and Rozenberg, G., editors, *Handbook of Graph Grammars and Computing by Graph Transformations, vol. 2: Applications, Languages and Tools*. World Scientific, 1999.

[11] Formisano, A. and Omodeo, E. G. (2000). An equational re-engineering of set theories. In Caferra, R. and Salzer, G., editors, *Automated Deduction in Classical and Non-Classical Logics*, LNCS 1761 (LNAI), pages 175–190. Springer.

[12] Formisano, A., Omodeo, E. G., and Simeoni, M. (2001). A graphical approach to relational reasoning. In Kahl, W., Parnas, D. L., and G. Schmidt, G., editors, *Electronic Notes in Theoretical Computer Science*, 44(3). Elsevier Science Publishers.

[13] Formisano, A., Omodeo, E. G., and Temperini, M. (1998). Plan of activities on the map calculus. In Freire-Nistal, J., Falaschi, M., and Vilares-Ferro, M., editors, *Proceedings of the AGP98 Joint Conference on Declarative Programming*, pages 343–356, A Coruña, Spain.

[14] Formisano, A., Omodeo, E. G., and Temperini, M. (2000). Goals and benchmarks for automated map reasoning. *Journal of Symbolic Computation*, 29(2):259–297.

[15] Formisano, A., Omodeo, E. G., and Temperini, M. (2001). Layered map reasoning: An experimental approach put to trial on sets. In Dovier, A., Meo, M.-C., and Omicini, A., editors, *Electronic Notes in Theoretical Computer Science*, 48:1–28. Elsevier Science Publishers.

[16] Formisano, A., Omodeo, E. G., and Temperini, M. (2001). Instructing Equational Set-Reasoning with Otter. In Goré, R., Leitsch, A., Nipkow, T., editors, *Automated Reasoning*, LNCS 2083 (LNAI), pages 152–167. Springer.

[17] Formisano, A. and Simeoni, M. (2001). An AGG application supporting visual reasoning. In Baresi, L., and Pezzè, M., editors, *Electronic Notes in Theoretical Computer Science*, 50(3). Elsevier Science Publishers.

[18] Frias, M. F. and Orlowska, E. (1995). A proof system for fork algebras and its applications to reasoning in logics based on intuitionism. *Logique & Analyse*, 150-151-152:239–284.

[19] Halmos, P. R. (1962). *Algebraic Logic*. Chelsea, New York.

[20] Halmos, P. R. and Givant, S. (1998). *Logic as Algebra*, volume 21 of *Dolciani Mathematical Expositions*. The Mathematical Association of America, Washington, D.C.

[21] Hilbert, D. (1976). Mathematical problems (lecture delivered before the international congress of mathematicians at Paris in 1900). In Browder, F. E., editor, *Mathematical developments arising from Hilbert problems (Symposium in pure mathematics, Northern Illinois University, 1974)*, volume 28 of *Proceedings of symposia in pure mathematics*, pages 1–34, Providence, Rhode Island, 1976. American Mathematical Society.

[22] Krivine, J. L. (1971). *Introduction to axiomatic set theory*. Reidel, Dordrecht. Holland.

[23] Kwatinetz, M. K. (1981). *Problems of expressibility in finite languages*. PhD thesis, University of California, Berkeley.

[24] Matiyasevich, Y. V. (1993). *Hilbert's Tenth Problem*. The MIT Press, Cambrige, Massachusetts, and London, England.

[25] Quaife, A. (1992). *Automated development of fundamental mathematical theories*. Kluwer Academic Publishers.

[26] Schmidt, G. and Ströhlein, T. (1993). *Relations and graphs*, Springer-Verlag, Monographs on Theoretical Computer Science.

[27] Shepherdson, J. C. (1984). Negation as failure: A comparison of Clark's completed data base and Reiter's closed world assumption. *Journal of Logic Programming*, 1(1):51–79.

[28] Szmielew, W. (1979). The role of the Pasch axiom in the foundations of Euclidean geometry. In Henkin, L., editor, *Proceedings of the Tarski Symposium*, volume XXV of *Proceedings of Symposia in Pure Mathematics*, pages 123–138. American Mathematical Society. Second printing.

[29] Tarski, A. (1941). On the calculus of relations. *Journal of Symbolic Logic*, 6(3):73–89.

[30] Tarski, A. (1969). What is elementary geometry? In Hintikka, J., editor, *The philosophy of mathematics*, Oxford readings in Philosophy, pages 164–175. Oxford University Press.

[31] Tarski, A. and Givant, S. (1987). *A formalization of Set Theory without variables*, volume 41 of *Colloquium Publications*. American Mathematical Society.

[32] Ullman, J. D. (1989). *Principles of Database and Knowledge-base Systems*. Principles of Computer Science. Computer Science Press, Stanford University.